Systême soft

Программный комплекс Систэм Платформ

SePlatform.HMI 2.0

Руководство пользователя

Редакция	Соответствует версии ПО
10. Предварительная	2.0.13

Содержание

1. Введение

В документе описаны основные принципы работы со средой разработки и исполнения визуальной части проектов автоматизации SePlatform.HMI. Документ предназначен для разработчиков мнемосхем и администраторов, которые занимаются организацией автоматизированных рабочих мест для сотрудников компании, работающих с программным комплексом Систэм Платформ.

Содержание данного документа предполагает, что читатель знаком с базовыми понятиями объектно-ориентированного подхода, имеет представление о работе в HMI[1] и IDE[2], проектах и о скриптовых языках, например, о JavaScript. Если читатель не знаком с данными понятиями, то документ рекомендуется читать последовательно: с начальных глав (описание базовых принципов работы в SePlatform.HMI) до финальных глав (сценарии применения SePlatform.HMI для решения конкретных задач).

Для удобства восприятия, в документе применяется цветовая индикация.

- Свойства компонентов выделяются голубым цветом:

ХостClosed

- События компонентов выделяются зеленым цветом:

MouseMove

- Функции компонентов выделяются синим цветом:

BeginUpdate()

Для детального ознакомления с полным перечнем свойств, событий и функций всех компонентов стандартной библиотеки, смотрите документ `SePlatform.HMI.Справочное руководство`.

[1] Human-machine interface (HMI) - программное решение, обеспечивающее взаимодействие человека-оператора с управляемыми им машинами.
[2] Integrated Development Environment (IDE) - интегрированная среда разработки.

2. О продукте

SePlatform.HMI - это среда разработки и исполнения визуальной части проектов автоматизации. Позволяет работать в двух режимах:

- Дизайнер SePlatform.HMI (стр. 14): для администраторов и разработчиков. Представляет собой интегрированную среду разработки (IDE) визуальных мнемосхем, используемых в проектах автоматизации производства;
- Визуализатор SePlatform.HMI (стр. 14): для операторов. Позволяет запускать на исполнение подготовленные с помощью дизайнера визуальные проекты автоматизации.

Как и другие программные продукты Систэм Платформ, SePlatform.HMI придерживается объектного подхода. Все элементы проекта являются объектами, которые могут взаимодействовать между собой - для этого в SePlatform.HMI есть богатый набор базовых функций, а также возможность расширения функциональности посредством использования сценариев на базе языка JavaScript или собственного скриптового языка SePlatform.Om.

Помимо стандартной библиотеки компонентов, дизайнер SePlatform.HMI позволяет разрабатывать собственные типы графических объектов (стр. 1) и многократно использовать их экземпляры в других проектах. Основная идея такого подхода заключается в возможности подготавливать пользовательские типы объектов и закладывать логику их поведения до формирования экранных форм. Экземпляры таких типов могут многократно внедряться в проект, что значительно снижает временные и трудовые ресурсы, а также обеспечивает гибкую настройку.

В текущей версии продукта сложные графические объекты подготавливаются пользователем самостоятельно. В дальнейшем планируется расширение библиотеки объектов и наполнение её готовыми типовыми конструкциями.

2.1. Системные требования

ОС Windows

ОС	Microsoft Windows 10 Pro/11 Pro Microsoft Windows Server 2012/2012 R2/2016/2019/2022
Разрядность ОС	x64
Процессор	Intel Celeron с тактовой частотой не менее 1.6 ГГц
Объем оперативной памяти	Не менее 1 ГБ
Объем дисковой памяти	Не менее 500 МБ
Сетевой адаптер	Ethernet 10/100/1000 Мбит/с

Установленное ПО	OPC Core Components Redistributable (x64) 105.1 https://opcfoundation.org/developer-tools/samples-and-tools-classic/core-components Microsoft Visual C++ 2015-2019 Redistributable (x86) https://www.microsoft.com/ru-ru/download/ Microsoft Visual C++ 2015-2019 Redistributable (x64) https://www.microsoft.com/ru-ru/download/ Microsoft .NET Framework 4.5.2 https://www.microsoft.com/ru-ru/download/details.aspx?id=49982

ОС Linux

ОС	Ubuntu (glibc не ниже 2.17), наличие GUI Astra Linux РЕД ОС
Разрядность ОС	x64
Процессор	Intel Celeron с тактовой частотой не менее 1.6 ГГц
Объем оперативной памяти	Не менее 1 ГБ
Объем дисковой памяти	Не менее 500 МБ
Сетевой адаптер	Ethernet 10/100/1000 Мбит/с

2.2. Установка и удаление

ОС Windows

Для установки или удаления SePlatform.HMI (дизайнера SePlatform.HMI и визуализатора SePlatform.HMI - SePlatform.HMI.Viewer) используйте дистрибутив `seplatform.hmi.desktop-<lng>-<version>.<arch>.msi`.

ПРИМЕЧАНИЕ

В названии файла `<lng>` - это язык компонента, `<version>` - номер версии компонента, а `<arch>` - целевая процессорная архитектура.

ОС Linux

Для установки или удаления SePlatform.HMI используйте пакет `seplatform.hmi.desktop-<lng>-<version>.<arch>.rpm` или `seplatform.hmi.desktop-<lng>-<version>.<arch>.deb` в зависимости от пакетного менеджера ОС.

ПРИМЕЧАНИЕ

В названии пакета `<lng>` - это язык компонента, а `<version>` - номер версии компонента.

ОБРАТИТЕ ВНИМАНИЕ

- Команды установки/удаления нужно запускать с привилегиями суперпользователя **root**, находясь в папке с установочным пакетом.
- В командах удаления вместо полного имени пакета достаточно указать seplatform.hmi-desktop.

Пакетный менеджер YUM

Установка

```
yum install seplatform.hmi.desktop-<lng>-<version>.<arch>.rpm
```

Удаление

```
yum remove seplatform.hmi-desktop
```

Использование пакетного менеджера RPM

Установка

```
rpm -i seplatform.hmi.desktop-<lng>-<version>.<arch>.rpm
```

Удаление

```
rpm -e seplatform.hmi-desktop
```

Пакетный менеджер APT

Установка

```
apt install ./seplatform.hmi.desktop-<lng>-<version>.<arch>.deb или apt-get install
./seplatform.hmi.desktop-<lng>-<version>.<arch>.deb
```

Удаление

```
apt remove seplatform.hmi-desktop или apt-get remove seplatform.hmi-desktop
```

Пакетный менеджер DPKG

Установка

```
dpkg -i seplatform.hmi.desktop-<lng>-<version>.<arch>.deb
```

Удаление

```
dpkg -r seplatform.hmi-desktop
```

После завершения установки SePlatform.HMI возможны следующие варианты запуска приложения:

- запуск дизайнера SePlatform.HMI (стр. 14).
- запуск скомпилированного проекта (стр. 14) SePlatform.HMI.

3. Основные этапы работы с проектом

3.1. Запуск дизайнера

ОС Windows

Для запуска дизайнера SePlatform.HMI воспользуйтесь командой `Пуск → SePlatform → SePlatform.HMI`. Аналогичное действие можно выполнить, запустив исполняемый файл `seplatform.hmi.designer.exe`, расположенный в папке установки.

ОС Linux

Способы запуска:

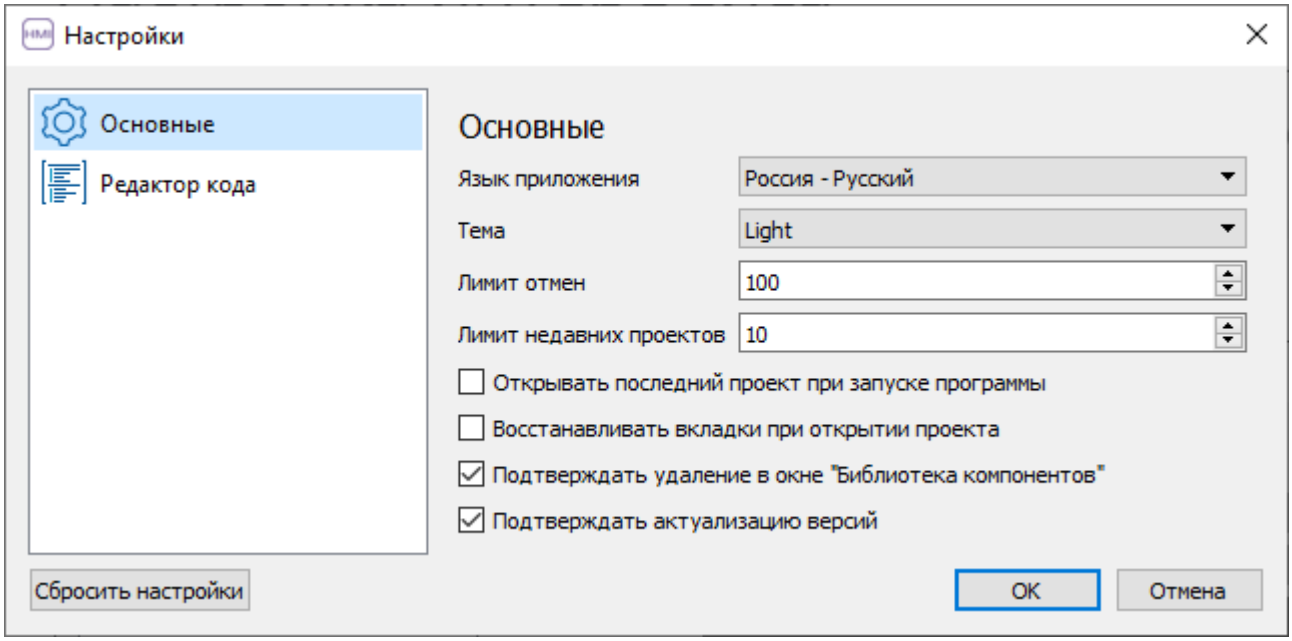
- командой `Пуск → Разработка → Дизайнер SePlatform.HMI`
- командой в терминале

```
seplatform.hmi.designer
```

3.1.1. Настройка интерфейса

Чтобы открыть окно настроек SePlatform.HMI, перейдите в меню **Файл** и выберите **Настройки**.

На вкладке **Основные** вы можете выбрать язык интерфейса, цветовую тему дизайнера или установить максимальное количество отменяемых действий в дизайнере.



3.1.2. Создание нового проекта

Чтобы создать новый проект, запустите дизайнер SePlatform.HMI и в меню **Файл** выберите **Создать проект** или нажмите сочетание клавиш **«Ctrl»+«N»**.

ПРИМЕЧАНИЕ

Каждый проект должен быть размещен в отдельной папке. Несоблюдение этого правила может привести к конфликтам и потере данных из-за перезаписи файлов.

Папка с проектом будет содержать в себе две папки - `objects` и `resources`, а также сам файл проекта с расширением `*.hmi`.

3.1.3. Доработка готового проекта

Для доработки готового проекта выполните одно из действий:

- в меню **Файл** выберите **Открыть проект** или нажмите сочетание клавиш **«Ctrl»+«O»**. В заголовке окна дизайнера будет отображен путь открытого проекта;
- перейдите в папку проекта и выберите команду **Изменить** (в ОС Windows) или **Открыть** (в ОС Linux) в контекстном меню файла проекта с расширением `*.hmi`.

3.1.4. Компиляция проекта

ОС Windows/ОС Linux

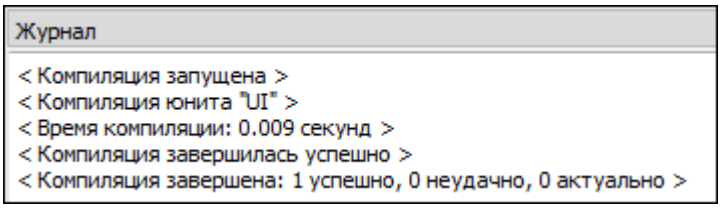
Компиляция формы, типа или всего проекта выполняется вручную либо автоматически при запуске в рантайм (стр. 14) формы, типа или проекта.

Чтобы вручную скомпилировать отдельную форму или тип, откройте нужную вкладку, перейдите в меню **Проект** и выберите **Запуск компиляции текущего типа** или нажмите сочетание клавиш **«Ctrl»+«F5»**. При компиляции вкладки компилируются также зависимые от нее формы и типы.

Чтобы скомпилировать весь проект, в меню **Проект** выберите **Запуск компиляции** или нажмите клавишу **«F5»**.

Чтобы прервать компиляцию, выберите команду **Проект** → **Остановить компиляцию** или нажмите сочетание клавиш **«Ctrl»+«Break»**. Команда отмены появляется в меню только во время компиляции.

Весь процесс, а затем результат компиляции отображаются в области **Журнал**.



Скомпилированный проект сохраняется в папку проекта в файл формата `*.binom`. Файл `Имя проекта.binom` позволяет очень быстро запустить проект в режиме рантайма без компиляции (стр. 18).

ОБРАТИТЕ ВНИМАНИЕ

Файл `*.binom` привязан к компьютеру, на котором он был создан. После переноса проекта на другой компьютер, скомпилируйте проект.

3.1.5. Сохранение

Чтобы сохранить изменения на активной вкладке, выполните одно из действий:

- перейдите в меню **Файл** и выберите команду **Сохранить**;
- нажмите кнопку **Сохранить** на **Панели инструментов**;
- нажмите сочетание клавиш **«Ctrl»+«S»**.

Чтобы сохранить все изменения проекта, выполните одно из действий:

- перейдите в меню **Файл** и выберите команду **Сохранить проект**,
- нажмите кнопку **Сохранить проект** на **Панели инструментов**;
- нажмите сочетание клавиш **«Ctrl»+«Shift»+«S»**.

Чтобы сохранить проект по указанному пути, выберите команду **Сохранить проект как...** в меню **Файл**.

Каждые 30 секунд в папку проекта `objects` автоматически сохраняется резервная копия проекта с расширением `*.bak`. Резервная копия удаляется при сохранении проекта или отклонении сохранения во время закрытия дизайнера SePlatform.HMI.

3.2. Запуск на исполнение

Проекты автоматизации, созданные и скомпилированные с помощью дизайнера SePlatform.HMI, запускаются на исполнение с помощью визуализатора SePlatform.HMI (SePlatform.HMI.Viewer).

3.2.1. Запуск из режима разработки

Из дизайнера SePlatform.HMI можно запустить:

- отдельную форму проекта, для этого выполните одно из действий:
  - перейдите в контекстное меню нужной формы в **Библиотеке компонентов** и выберите **Показать в рантайме**;
  - откройте нужную форму, перейдите в меню **Проект** и выберите **Показать активную вкладку в рантайме**;
  - откройте нужную форму и нажмите сочетание клавиш **«Ctrl»+«F9»**;
- главную экранную форму (стр. 262), для этого перейдите в меню **Проект** и выберите **Показать главную форму в рантайме** или нажмите клавишу **«F9»**.

> ПРИМЕЧАНИЕ
>
> При запуске в рантайм отдельной формы, типа или объекта выполняется компиляция и проверка ошибок только выбранной формы, типа или объекта и зависимых форм, типов и объектов. При запуске главной формы компилируется и проверяется весь проект.

3.2.2. Запуск из командной строки (ОС Windows)/терминала (ОС Linux)

3.2.2.1. Запуск с компиляцией

Чтобы запустить в режим рантайма форму, тип или проект с автоматической компиляцией, следует использовать для запуска файл проекта в формате `*.hmi`.

ПРИМЕЧАНИЕ

При запуске в рантайм отдельной формы или типа выполняется компиляция и проверка ошибок только выбранной формы или типа и зависимых форм и типов. При запуске главной формы компилируется и проверяется весь проект.

ОС Windows

Перед запуском в рантайм отдельной формы, типа или всего проекта перейдите в папку установки SePlatform.HMI.

ПРИМЕР

```
cd C:\"Program Files"\SePlatform\SePlatform.HMI
```

Чтобы запустить в рантайм отдельную форму, тип или весь проект, выполните команду

```
./seplatform.hmi.viewer.exe <полный путь к .hmi файлу> <имя формы> <имя типа/объекта>
```

ОБРАТИТЕ ВНИМАНИЕ

Если в пути к `.hmi` файлу или в имени экранной формы присутствуют пробелы, используйте двойные кавычки.

ПРИМЕЧАНИЕ

Для запуска главной формы (стр. 262) имя формы можно не указывать.

ПРИМЕР

Запуск на исполнение типа:

```
./seplatform.hmi.viewer.exe C:\Projects\HMI.Projects\TU_402.hmi SW
```

ПРИМЕР

Запуск на исполнение проекта (главной формы):

```
./seplatform.hmi.viewer.exe C:\Projects\HMI.Projects\TU_402.hmi
```

ПРИМЕР

Запуск на исполнение формы с пробелами в названии:

```
./seplatform.hmi.viewer.exe C:\Projects\HMI.Projects\TU_402.hmi "Tank date"
```

ОС Linux

Чтобы запустить в рантайм отдельную форму, тип или весь проект, находясь в директории с проектом, выполните команду

```
./seplatform.hmi.viewer.exe <имя проекта>.hmi <имя формы> <имя типа/объекта>
```

ПРИМЕЧАНИЕ

Для запуска главной формы (стр. 262) значение параметра имя формы можно не указывать.

ПРИМЕР

Запуск на исполнение типа:

```
./seplatform.hmi.viewer.exe TU_402.hmi SW
```

ПРИМЕР

Запуск на исполнение проекта (главной формы):

```
./seplatform.hmi.viewer.exe TU_402.hmi
```

Чтобы запустить в рантайм отдельную форму, тип или весь проект, находясь вне директории с проектом, используйте вышеописанные команды и укажите полный путь к проекту

ПРИМЕР

Запуск на исполнение проекта (главной формы):

```
./seplatform.hmi.viewer.exe Projects/HMI.Projects/TU_402.hmi
```

3.2.2.2. Запуск без компиляции

Чтобы запустить в режим рантайма форму, тип, или весь проект без компиляции, следует использовать для запуска файл скомпилированного проекта (стр. 15) в формате `*.binom`.

ВАЖНО

Файл `*.binom` привязан к версии SePlatform.HMI. Запускайте проект только при наличии той же версии, на которой был создан файл.

ОС Windows/ОС Linux

Команды запуска в рантайм формы, типа или проекта без компиляции аналогичны запуску с компиляцией (стр. 16). Вместо формата `*.hmi` следует указывать формат `*.binom`.

ПРИМЕР

Запуск на исполнение проекта (главной формы) в ОС Windows:

```
./seplatform.hmi.viewer.exe C:\Projects\HMI.Projects\TU_402.binom
```

ПРИМЕР

Запуск на исполнение проекта (главной формы) в ОС Linux:

```
./seplatform.hmi.viewer.exe TU_402.binom
```

3.2.3. Запуск из проводника Windows

Чтобы запустить проект в рантайм, перейдите в папку проекта и два раза щелкните по файлу проекта с расширением `*.hmi`. В результате будет запущена форма, указанная главной (стр. 262).

ПРИМЕЧАНИЕ

При запуске выполняется компиляция главной формы и зависимых от нее форм, типов и объектов.

3.2.4. Запуск визуализатора с параметрами

Чтобы отобразить справку, введите ключи -?, -h или --help.

ПРИМЕР

Вызов окна справки:

```
./seplatform.hmi.viewer.exe --help
```

Чтобы отобразить информацию о версии визуализатора, используйте ключи -v, --version.

ПРИМЕР

Вызов окна информации о версии:

```
./seplatform.hmi.viewer.exe --version
```

Чтобы отключить сглаживание изображений в визуализаторе, используйте ключи --disable AA, --disable antialiasing.

ПРИМЕР

```
./seplatform.hmi.viewer.exe C:\MyProjects\MyProject.hmi New_Form --disable AA
```

3.3. Обновление проекта без перезапуска

Во время исполнения проекта (в визуализаторе SePlatform.HMI) можно редактировать проект в Дизайнере SePlatform.HMI и тут же подгружать изменения в визуализатор. Для подгрузки обновлений в запущенный проект добавьте в любое место проекта экземпляр типа **Менеджер проекта**.

Благодаря типу **Менеджер проекта** вы можете запустить проект в режим исполнения на одном рабочем месте и дорабатывать проект на другом, обновляя запущенный проект по мере доработок. Для работы типа **Менеджер проекта** подготовьте две пустые папки:

- папка для обновлений - в эту папку нужно помещать файлы проекта с изменениями, если вы хотите подгрузить в запущенный проект эти изменения. Тип **Менеджер проекта** следит за содержимым этой папки и заменяет файлы текущего проекта файлами из неё;
- папка для автосохранения копии запущенного проекта - в эту папку при каждом обновлении проекта автоматически сохраняется копия текущего запущенного проекта для возможности отката изменений.

3.3.1. Подгрузка данных в запущенный проект

Чтобы подгружать обновления в запущенный проект, используйте функцию AsyncReload элемента *ProjectManager_1* типа **Менеджер проекта**.

Чтобы настроить подгрузку обновлений проекта, выполните:

1. Выберите обработчик события, при возникновении которого следует выполнять подгрузку данных и пропишите в коде обработчика вызов функции AsyncReload. В параметрах функции AsyncReload элемента *ProjectManager_1* пропишите путь к изменённому проекту и папке автосохранения копии проекта.

ПРИМЕР

```
ProjectManager_1.AsyncReload("C:\\WORK\\HMI\\PROJECT_update\\PROJECT.hmi",
"C:\\WORK\\HMI\\PROJECT_backup");
```

2. Запустите проект с помощью Визуализатора SePlatform.HMI (перейдите в контекстное меню файла с расширением `.hmi` и выполните команду **Запустить**).
3. Откройте проект в Дизайнере SePlatform.HMI и внесите изменения.
4. Скопируйте все файлы проекта и подложите их в папку обновлений (которая может находиться на другом рабочем месте). Таким образом следует подкладывать файлы в папку обновлений каждый раз после изменения проекта в дизайнере.
5. Для подгрузки обновлений вызывайте функцию AsyncReload в запущенном проекте.

3.3.2. Проверка наличия обновлений

Чтобы проверить наличие файлов с изменениями в папке обновлений, используйте функцию CheckForUpdates элемента *ProjectManager_1* типа **Менеджер проекта**.

Выберите обработчик события, при возникновении которого следует выполнять проверку обновлений и пропишите в обработчике вызов функции CheckForUpdates. В параметрах функции CheckForUpdates элемента *ProjectManager_1* пропишите путь к обновлённому проекту.

ПРИМЕР

```
Text_4 = ProjectManager_1.CheckForUpdates("C:\\WORK\\HMI\\PROJECT_
update\\PROJECT.hmi") ? "true" : "false";
```

3.3.3. Откат изменений

Чтобы вернуть проект к исходному виду (до подгрузки последних обновлений), используйте функцию AsyncReloadFromBackup элемента *ProjectManager_1* типа **Менеджер проекта**.

Выберите обработчик события, при возникновении которого следует выполнять откат изменений и пропишите в обработчике вызов функции AsyncReloadFromBackup. В параметрах функции AsyncReloadFromBackup элемента *ProjectManager_1* пропишите путь к папке автосохранения копии проекта.

ПРИМЕР

```
ProjectManager_1.AsyncReloadFromBackup("C:\\WORK\\HMI\\PROJECT_backup");
```

4. Разработка визуальной части проекта

4.1. Работа с объектами в рабочей области

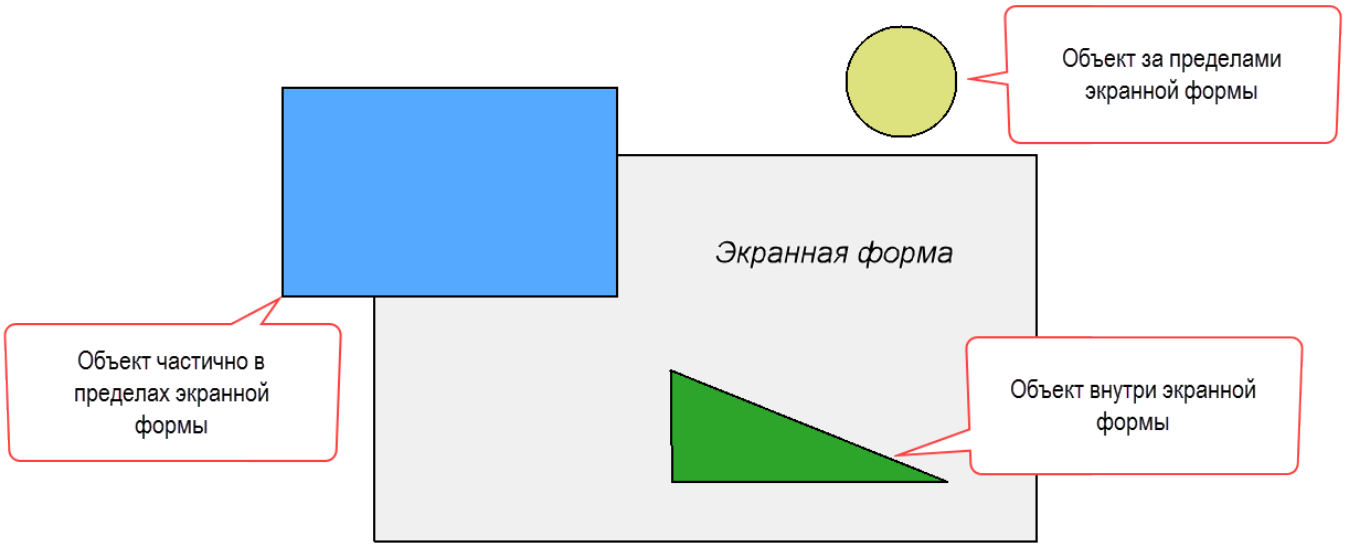
4.1.1. Добавление объектов на форму

Чтобы приступить к построению мнемосхемы, создайте проект (стр. 14) с экранной формой. Чтобы добавить объекты:

- перетащите нужные компоненты из библиотеки на рабочую область или в область **Структура объекта**;
- в **Структуре объекта** выберите объект, которому нужно добавить дочерний, откройте контекстное меню и выберите **Создать**.

Для объектов, размещенных на рабочей области, действуют стандартные операции форматирования **Вырезать** (**«CTRL»+«X»**), **Копировать** (**«CTRL»+«C»**) и **Вставить** (**«CTRL»+«V»**). Эти команды можно вызвать с панели инструментов, выбрать в меню **Правка** или в контекстном меню объекта в области **Структура объекта**.

Проектируя визуальную часть проекта, помните, что после запуска в рантайме (стр. 16) в окне экранной формы отобразятся только те объекты, которые были расположены в пределах экранной формы.



4.1.2. Именование объектов

Чтобы переименовать добавленный объект, перейдите в область **Структура объекта** и дважды щелкните по имени объекта либо вызовите контекстное меню объекта и выберите команду **Переименовать**.

ОБРАТИТЕ ВНИМАНИЕ

Следите, чтобы имена объектов в папках и вне папок не совпадали. При обращении к объектам приоритет всегда отдается объектам вне папок.

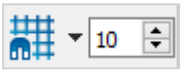
ОБРАТИТЕ ВНИМАНИЕ

Чтобы обратиться к объекту, расположенному в папке, в коде указывайте только имя объекта без имени папки.

Для любого объекта или элемента вы можете указать краткое русскоязычное описание в свойстве **Отображаемое имя**. При этом сохраняется возможность обращения к объекту/элементу в скриптах по англоязычному имени, которое указано в области **Структура объекта**. Удобно использовать отображаемое имя для форм, типов (стр. 56), экземпляров типов (стр. 56) и полей.

4.1.3. Выравнивание и распределение

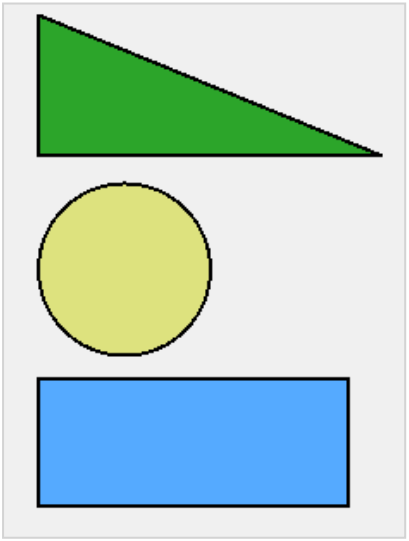
Чтобы выровнять объекты вручную, включите привязку к сетке на **Панели инструментов**. В поле ввода можно установить ширину шага (в пикселях).

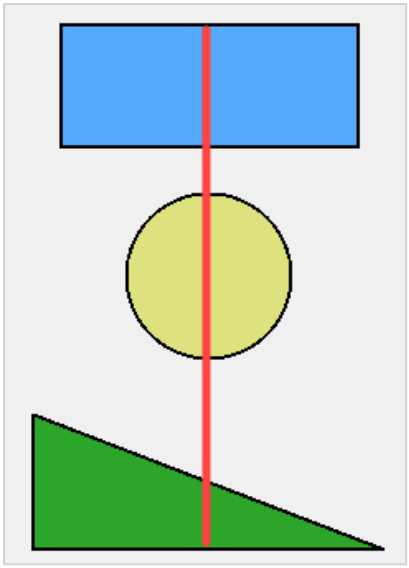


Чтобы выровнять группу объектов автоматически, используйте инструменты выравнивания **Панели инструментов**.

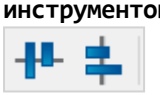
Чтобы выровнять объекты по краю (левый, правый, верхний, нижний) или центру, выделите нужную группу объектов и примените один из инструментов выравнивания **Панели инструментов**.

Ниже показаны результаты выравнивания трех объектов по левому краю, по центру и по нижнему краю.

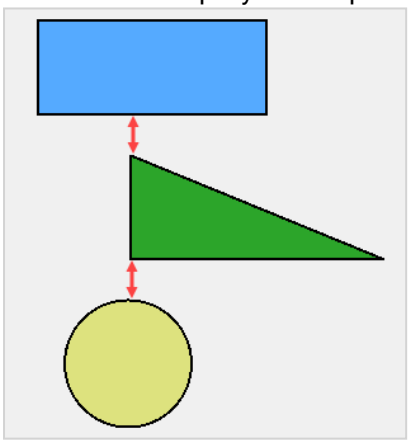


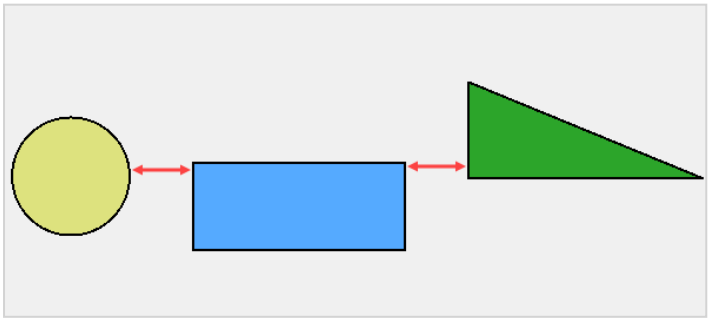


Чтобы равномерно распределить объекты на рабочей области (с одинаковыми зазорами), выделите нужную группу объектов и используйте инструменты распределения (по горизонтали, по вертикали) на **Панели инструментов**.

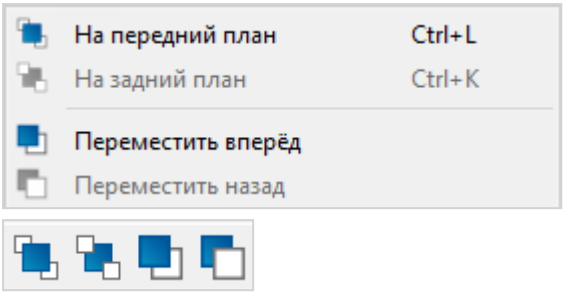


Ниже показаны результаты равномерного распределения объектов по вертикали и горизонтали.

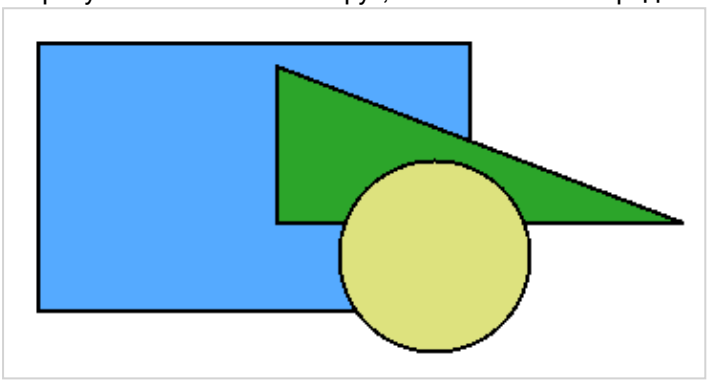


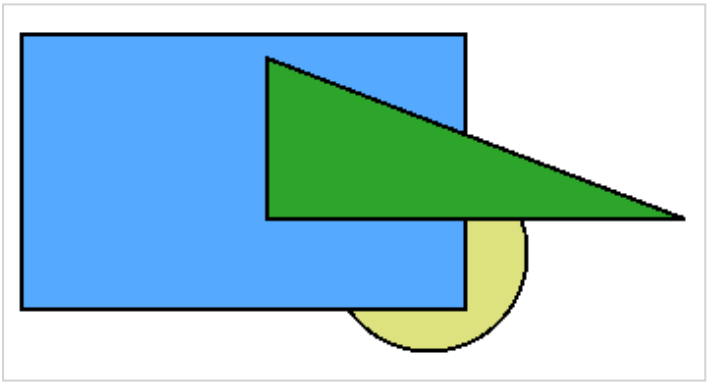


Чтобы изменить порядок объектов, перекрывающих друг на друга, выделите нужный объект и укажите нужный порядок для объекта в контекстном меню, либо используя кнопки **Панели инструментов**.



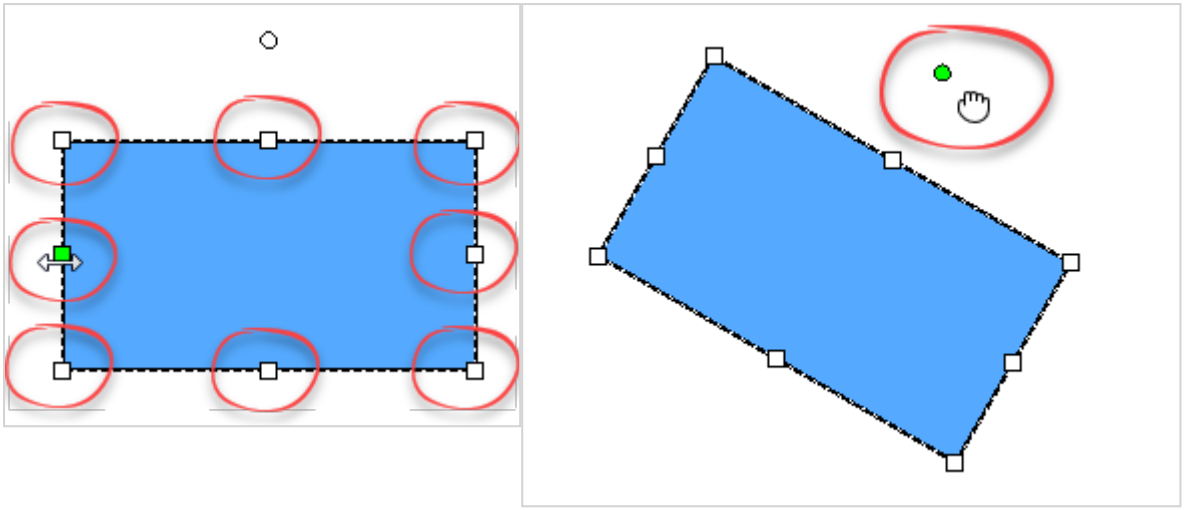
Комбинации **«Ctrl»+«L»**, **«Ctrl»+«K»** используются для быстрого перемещения объекта на передний или задний план.
На рисунках ниже показан круг, вынесенный на передний план и на задний план.



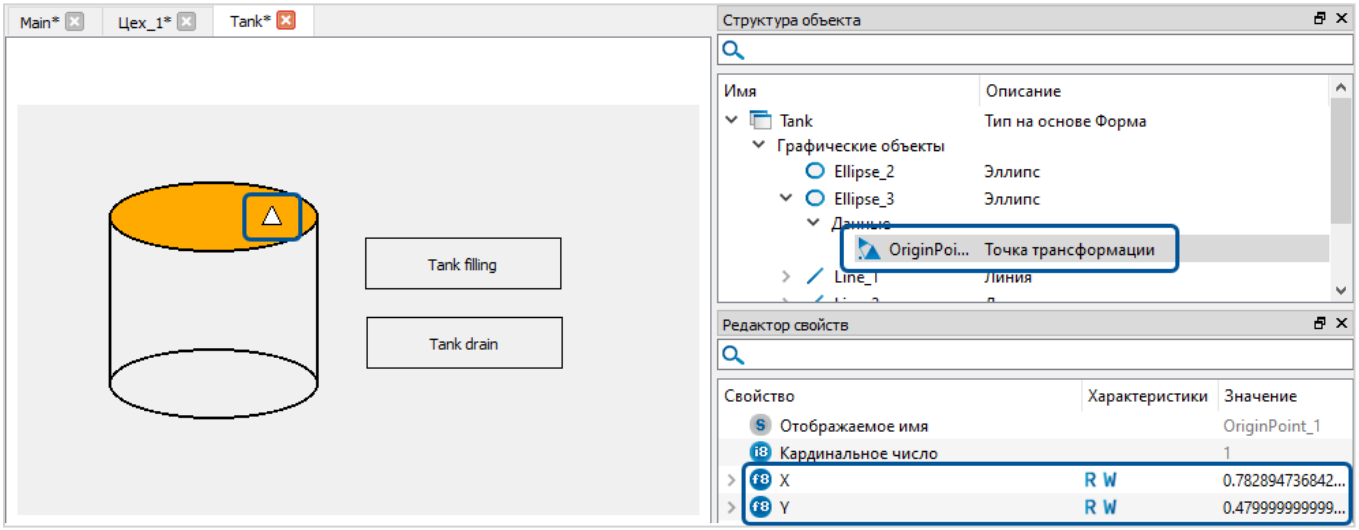


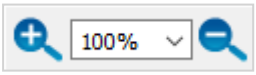
4.1.4. Масштабирование и поворот

Объекты, добавленные на рабочую область, можно растягивать или поворачивать, используя маркеры захвата. Менять размер объекта можно также с помощью сочетаний клавиш **«Ctrl»+«↓»**, **«Ctrl»+«↑»**, **«Ctrl»+«←»**, **«Ctrl»+«→»**. Чтобы изменить размер объекта без нарушения пропорций, растягивайте его с зажатой клавишей **«SHIFT»**.

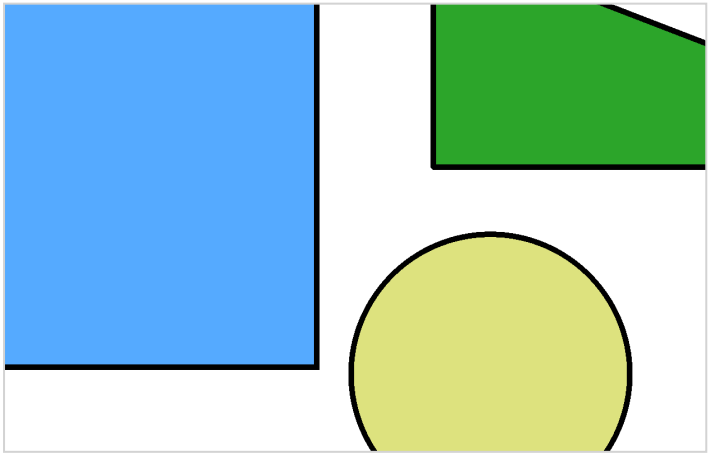


Чтобы изменить точку, относительно которой происходит масштабирование или поворот объекта, добавьте на объект компонент библиотеки **Точка трансформации** и укажите координаты точки трансформации (свойства X и Y). Перемещать точку трансформации можно также с помощью мыши, либо клавиш «↓», «↑», «←», «→».



Чтобы масштабировать рабочую область и все объекты рабочей области, используйте инструменты панели масштабирования 

или поворачивайте колесико мыши с зажатой клавишей **«CTRL»**. Ниже показан результат масштабирования рабочей области.



ОБРАТИТЕ ВНИМАНИЕ

Чтобы включить/отключить сглаживание изображения при работе в дизайнере SePlatform.HMI, используйте функцию **Сглаживание изображений**, которая находится в меню **Вид**.

ОБРАТИТЕ ВНИМАНИЕ

Чтобы отключить сглаживание изображений в визуализаторе SePlatform.HMI (в визуализаторе сглаживание изображений включено по умолчанию), запустите визуализатор (стр. 16) с параметром --disable AA.

4.1.5. Управление видимостью и активностью

Чтобы выключить видимость объекта в режимах разработки и исполнения, установите свойство **Видимость** в значение **«false»**.

Чтобы деактивировать объект (в режимах разработки и исполнения элемент будет виден, но не будет реагировать на действия пользователя), установите свойство **Включено** в значение **«false»**.

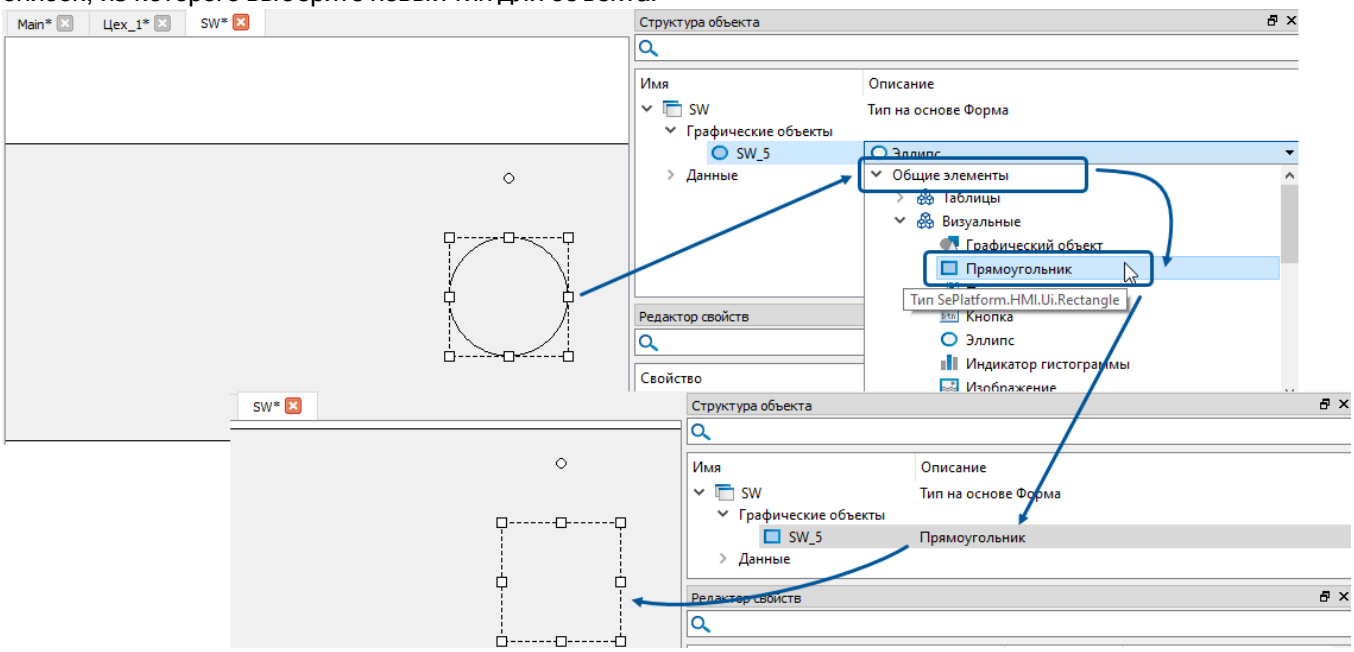
Чтобы управлять видимостью/активностью всех объектов экранной формы, меняйте значения свойств **Видимость**/**Включено** экранной формы.

При деактивации объекта деактивируются все его дочерние объекты. При активации родительского объекта активируются только те дочерние объекты, которые были активны до выключения родителя.

4.1.6. Изменение типа

Через структуру объекта можно быстро менять тип объекта. Смена допустима в рамках схожих типов (например, объект типа **Эллипс** на объект типа **Прямоугольник**, или элемент типа **OPC float4** на объект типа **OPC float8**).

Чтобы изменить тип, в структуре выберите объект и нажмите на тип. В результате появится выпадающий список, из которого выберите новый тип для объекта.



Информация об изменении типа объекта отражается в истории изменений (стр. 276).

4.1.7. Перемещение объектов в другие объекты

Вложенность объектов удобно использовать для распространения изменения свойств объекта-родителя на параметры дочерних объектов. Чтобы вложить объекты друг в друга, перетаскивайте их мышью в области **Структура объекта**. Визуальные графические объекты автоматически перемещаются на рабочей области в соответствии с иерархией в **Структуре объектов**. Привязку невизуальных графических объектов к объекту-родителю можно отследить только в **Структуре объектов**.

ПРИМЕЧАНИЕ

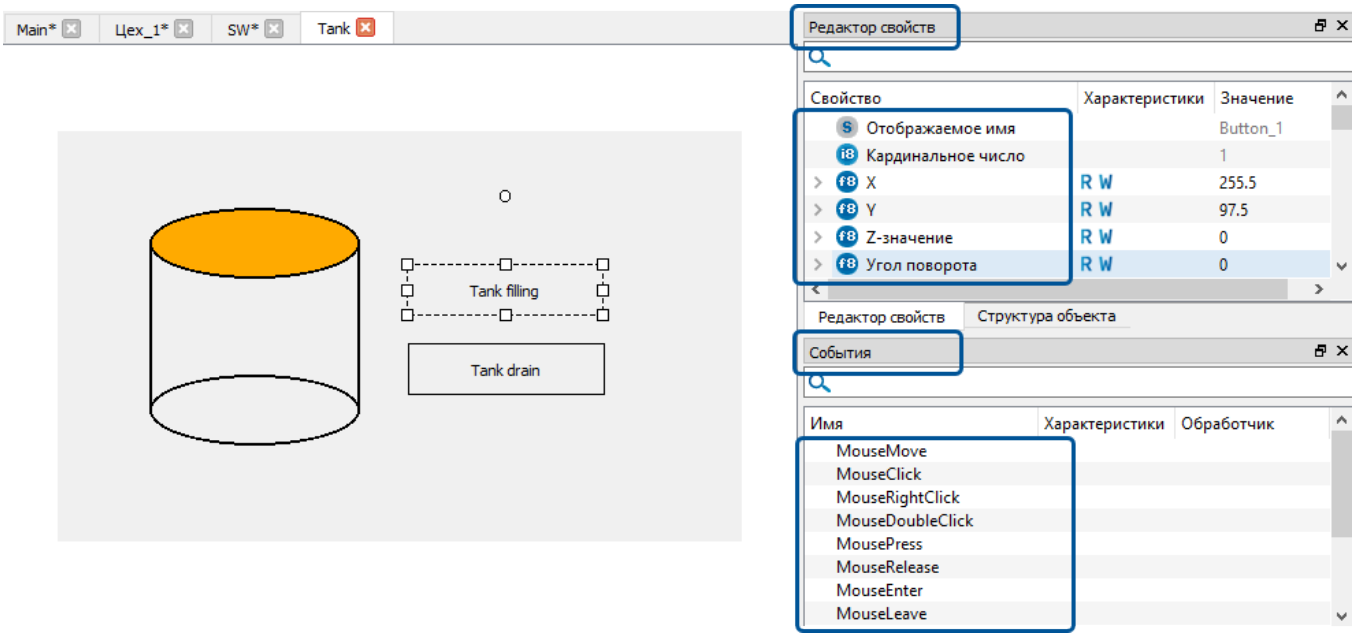
Для быстрой навигации в **Структуре объектов** используйте кнопки клавиатуры ←↑→↓.

4.1.8. Обзор характеристик

Любой объект проекта характеризуется свойствами (стр. 66), функциями (стр. 99) и событиями (стр. 85).

Свойства - это атрибуты, которые определяют особенности поведения или отображения объекта на экране (например, **ширина**, **высота** или **цвет заливки** для графического объекта). Свойства выделенного объекта отображаются в области **Редактор свойств**. Пользователь может создавать свои собственные свойства для любых объектов.

Чтобы реагировать на возникновение различных ситуаций у объектов существует понятие события. Таким образом, чтобы приложение выполняло некоторую работу в ответ на некое событие, разработчик должен определить реакцию на возникновение события - обработчик события (стр. 85) (например, обработчик на событие двойного клика мыши по объекту MouseDoubleClick). Любое событие может иметь параметры, которые его характеризуют (например, для события двойного клика мыши по объекту, параметрами будут выступать локальные координаты X и Y, где был произведен этот клик). Все события выделенного объекта можно посмотреть в области **События**.



Функции - вызываемые процедуры, позволяющие влиять на состояние объектов или их свойств. Как правило, у функций есть набор входных параметров. К примеру, у объекта стандартной библиотеки компонентов **Выпадающий список** есть предустановленная функция (стр. 99) RemoveItem(Index), предназначенная для удаления элемента списка с номером Index.

В SePlatform.HMI можно определять свои собственные функции (стр. 99) для любых объектов, используя язык SePlatform.Om или JavaScript. Любая созданная функция может возвращать значение, обращаться к входным аргументам в функции, а также вызывать другие функции.
Чтобы подробно ознакомиться с любыми характеристиками (свойства, события или функции) конкретных компонентов, см. документ `SePlatform.HMI.Справочное руководство`.

4.1.9. Поиск объектов на рабочей области

Чтобы найти объекты на рабочей области, выберите команду **Правка** → **Найти** или нажмите **«CTRL»+«F»**. В появившемся окне **Поиск по проекту** введите название объекта и нажмите кнопку **Найти**.

Чтобы при поиске учитывался регистр символов искомой фразы, отметьте флаг **С учётом регистра**.

Для поиска могут применяться символы подстановки:

- **«*»** - заменяют любую строку символов;
- **«?»** - заменяет любой одиночный символ.

Примеры:

ПРИМЕР

Найти в проекте объекты, в названиях которых есть сочетание символов "tu1":

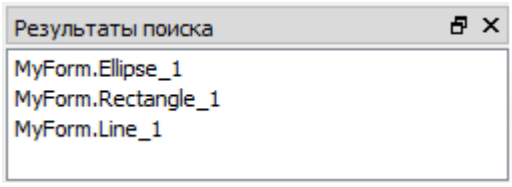
```
*tu1*
```

ПРИМЕР

Найти в проекте объекты, в названиях которых заканчивается на ".txt", а начало содержит любые 3 символа:

```
???.txt
```

Перечень найденных объектов появляется в области **Результаты поиска**.

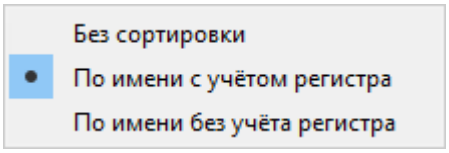


4.1.10. Фильтрация библиотеки компонентов

Чтобы отфильтровать компоненты в библиотеке, введите нужное сочетание символов в поле  . После фильтрации отобразятся только вложенные элементы групп и папок (компоненты, экранные формы, типы, глобальные объекты). Родительские элементы (группы, папки) отобразятся только если один или несколько их вложенных компонентов соответствуют фильтру.

4.1.11. Сортировка элементов в библиотеке компонентов

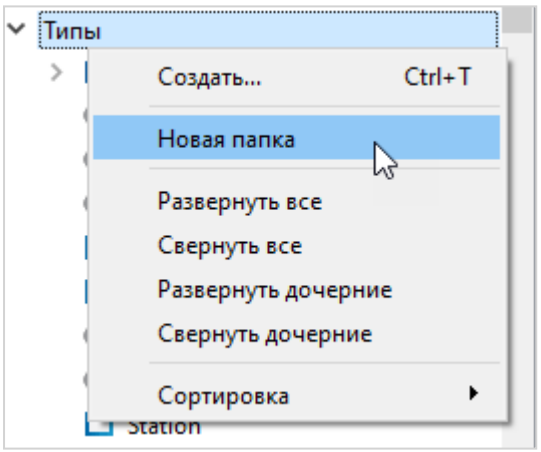
Чтобы отсортировать элементы в библиотеке компонентов, вызовите контекстное меню, выберите команду **Сортировка** и укажите тип сортировки. Сортировка применяется для всех групп библиотеки одновременно.

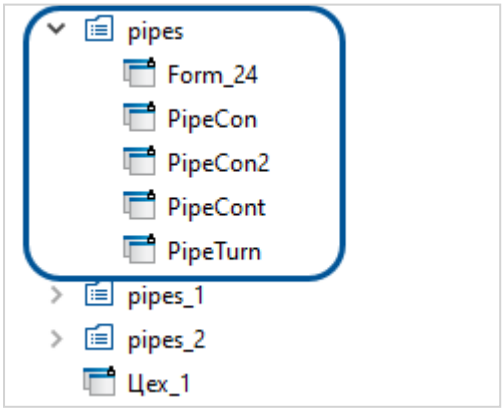


- **«Без сортировки»** - элементы располагаются в произвольном порядке;
- **«По имени с учетом регистра»** - элементы располагаются по алфавиту, в начало списка помещаются элементы, названия которых начинаются с заглавной буквы;
- **«По имени без учета регистра»** - элементы располагаются по алфавиту.

4.1.12. Группировка в библиотеке компонентов

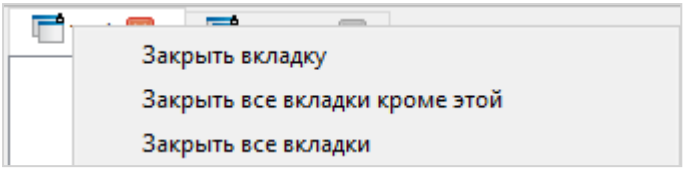
Чтобы сгруппировать экранные формы, типы или глобальные объекты, поместите их в папку. Чтобы создать папку, перейдите в контекстное меню элементов **Типы графических объектов/Экранные формы/Глобальные объекты** и выберите команду **Новая папка**.





4.1.13. Завершение редактирования

Чтобы завершить просмотр или редактирование объектов, типов или экранных форм, вызовите контекстное меню редактируемой вкладки и выберите нужную команду.



Для быстрого закрытия текущей вкладки используйте сочетание клавиш «Ctrl»+«W».

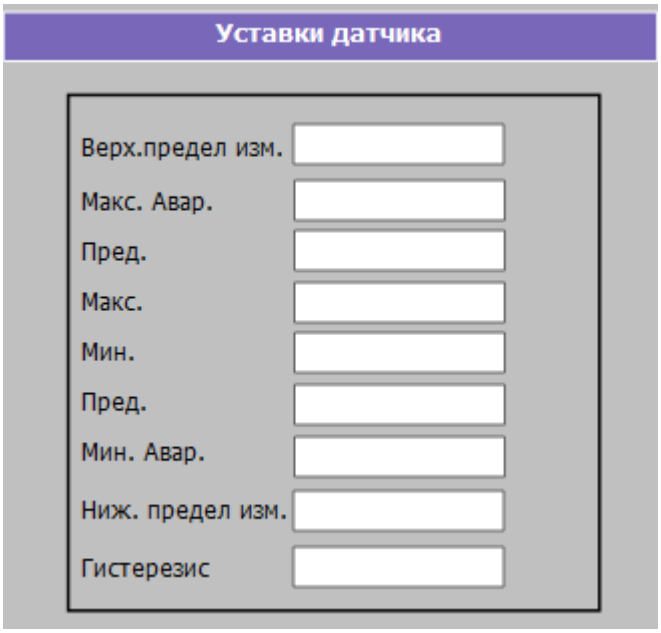
4.2. Использование графических примитивов

Компоненты графических примитивов (линии, прямоугольники, эллипсы и т.д.) используются в проекте автоматизации для рисования отдельных фигур и объектов технологического процесса (задвижки, резервуары, насосы и т.д.).

4.2.1. Прямоугольник

Компонент **Прямоугольник** применяется на мнемосхемах при рисовании объектов технологического процесса и, в некоторых случаях, как альтернатива Кнопке (стр. 43), так как имеет более широкие возможности для обработчиков событий (стр. 85). На рисунках ниже показаны экранные формы управления защитой и установки значений датчика. В первом случае прямоугольник выделяется рамкой, если на него наводится курсор мыши (срабатывает обработчик события MouseEnter). Во втором случае, если кликнуть мышью в область прямоугольника (срабатывает обработчик события MouseClick), откроется форма для более удобного ввода значений датчика.

1	Аварийная загазованность в БИК	М	Деблок.
2	Сохранение предельной загазованности в БИК	М	Деблок.
3	Недостоверность измерения двух датчиков загазованности в БИК	М	Деблок.
4	Аварийная загазованность в помещении ТПУ	М	Деблок.
5	Сохранение предельной загазованности в помещении ТПУ	М	Деблок.
6	Недостоверность измерения двух датчиков загазованности в ТПУ	М	Деблок.
7	Аварийный максимальный уровень затопления блок-бокса БИК	М	Деблок.
8	Аварийный максимальный уровень затопления ТПУ	М	Деблок.
9	Авар. макс. уровень в емкости сбора неучтенного нефтепродукта	М	Деблок.
10	Авар. макс. уровень в емкости сбора учтенного нефтепродукта	М	Деблок.
11	Пожар в помещении блок-бокса БИК	М	Деблок.
12	Пожар в помещении блок-бокса ТПУ	М	Деблок.



- цвет заливки прямоугольника устанавливается свойствами **Цвет заливки** и **Стиль заливки**;
- вид рамки прямоугольника устанавливается свойствами **Цвет пера**, **Стиль пера** и **Толщина пера**;
- скругление углов прямоугольника регулируется свойством **Радиус скругления**.

4.2.2. Эллипс

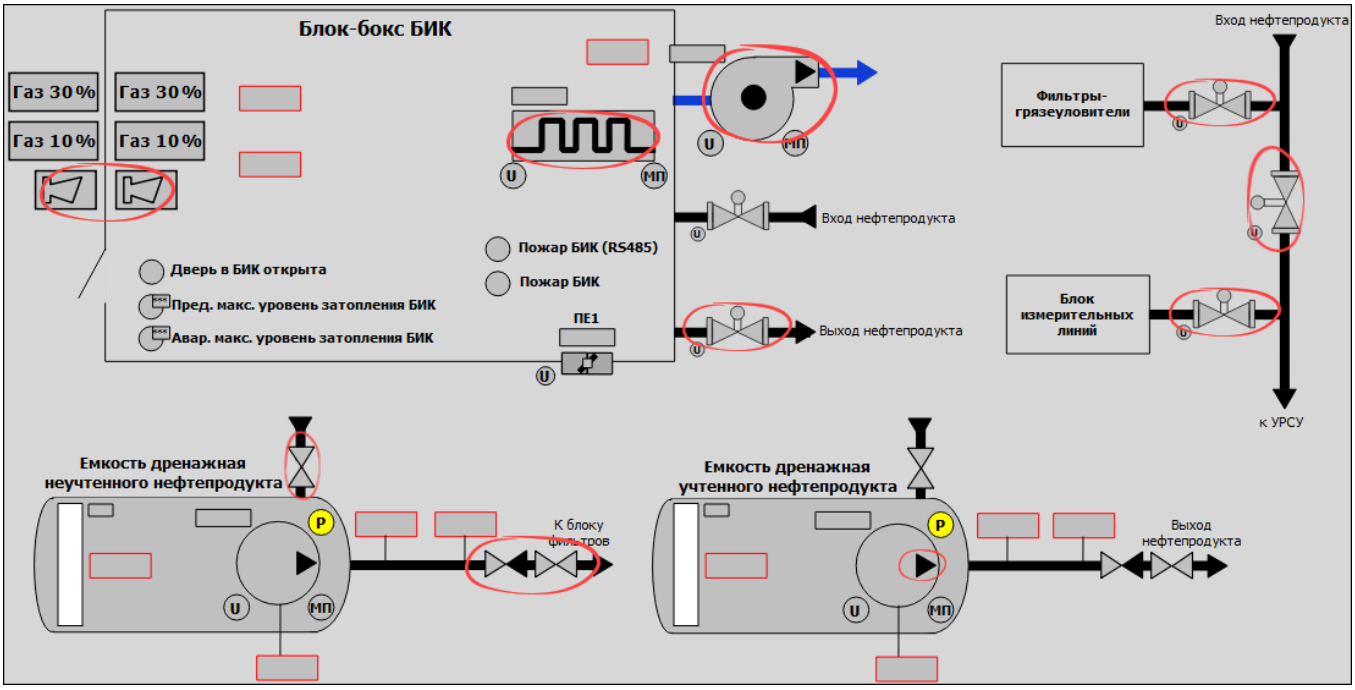
Компонент **Эллипс** применяется на мнемосхемах при рисовании объектов технологического процесса. На рисунке ниже показан участок мнемосхемы, где используется множество эллипсов для отражения состояния технологического объекта при наведении на них курсора (срабатывает обработчик события MouseEnter).

- вид границы эллипса определяется свойствами **Цвет пера**, **Стиль пера** и **Толщина пера**;
- вид заливки эллипса определяется свойствами **Цвет заливки** и **Стиль заливки**;
- всплывающее сообщение (при наведении курсора) устанавливается свойством **Всплывающая подсказка**.

4.2.3. Линия

Компонент **Линия** применяется на мнемосхемах при рисовании объектов технологического процесса. Помимо обычных ломаных линий, с помощью этого инструмента можно рисовать многоугольники (стр. 35). На

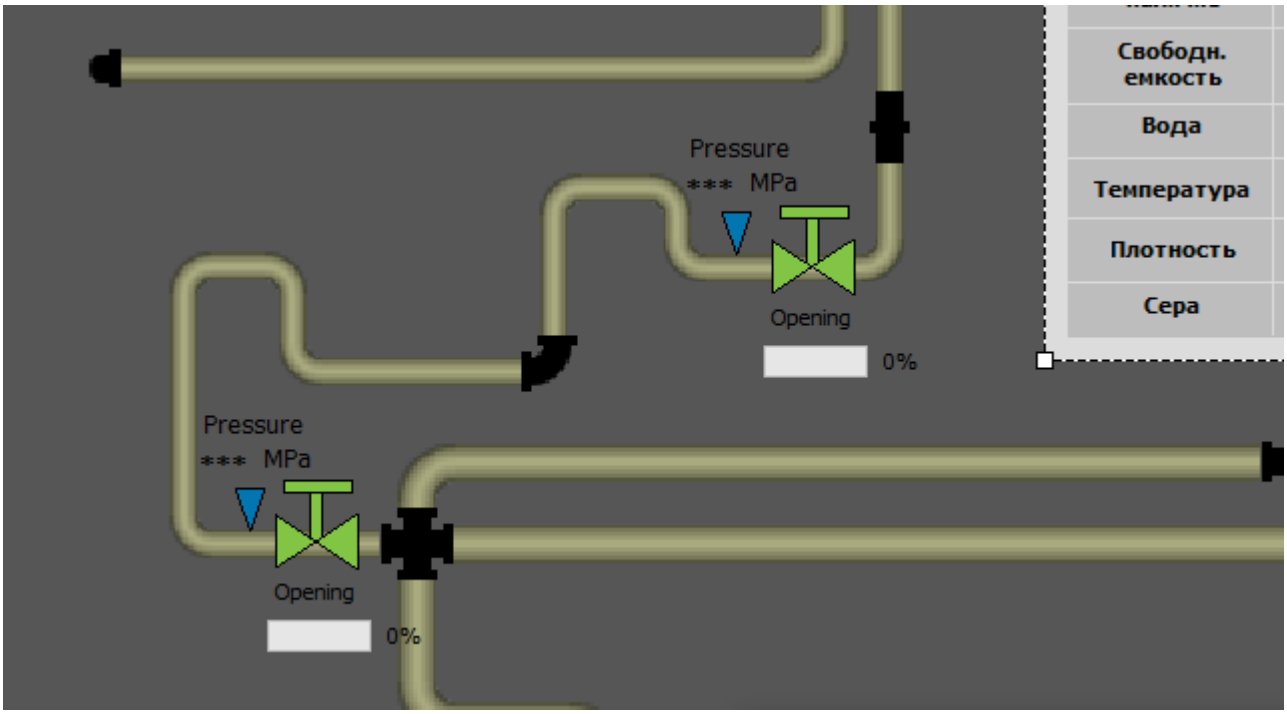
рисунке ниже показан участок мнемосхемы, где используется множество различных фигур, созданных с помощью компонента **Линия**.



Цвет, стиль и толщина линии устанавливается свойствами Цвет пера, Стиль пера и Толщина пера.

4.2.4. Соединительная линия и место соединения

Чтобы нарисовать сегменты трубопровода и соединители труб, используйте элементы **Соединительная линия** и **Место соединения**. Основное отличие элемента **Соединительная линия** от элемента **Линия** - возможность перемещения отдельных отрезков линии. Такая возможность позволяет построить сложный участок трубопровода из одной линии.



Рисование труб

Чтобы перемещать точки и отрезки между точками, перейдите в режим редактирования линии. Выделите линию, перейдите в контекстное меню и выберите команду **Редактировать линию** или нажмите клавишу **«F2»**. Чтобы выйти из режима редактирования, нажмите клавишу **«Esc»**.

Чтобы изменить внешний вид трубы, используйте свойства элемента **Соединительная линия**:

1. Включите декоратор линии в свойстве **Декоратор**. Декоратор позволяет добавить объемность в изображение трубы.
2. Задайте цвет линии в свойстве **Цвет пера**. Линия окрашивается в градиентный цвет при выборе любого цвета, кроме черного.

Чтобы менять толщину линии, используйте свойство **Масштаб**.

Рисование соединителей (мест соединения)

Чтобы выбрать тип и цвет места соединения, используйте свойства элемента **Место соединения**:

1. Включите декоратор в свойстве **Декоратор**. Декоратор позволяет изобразить соединитель в фигурном виде и добавить объемность в изображение.
2. Укажите тип места соединения в свойстве **Тип соединения**:

0	
1	
2	
3	
4	

3. Задайте цвет соединителя в свойстве **Цвет пера**. Соединитель окрашивается в градиентный цвет при выборе любого цвета, кроме черного.

Чтобы менять величину соединителя, используйте свойство Масштаб.

4.2.5. Многоугольник

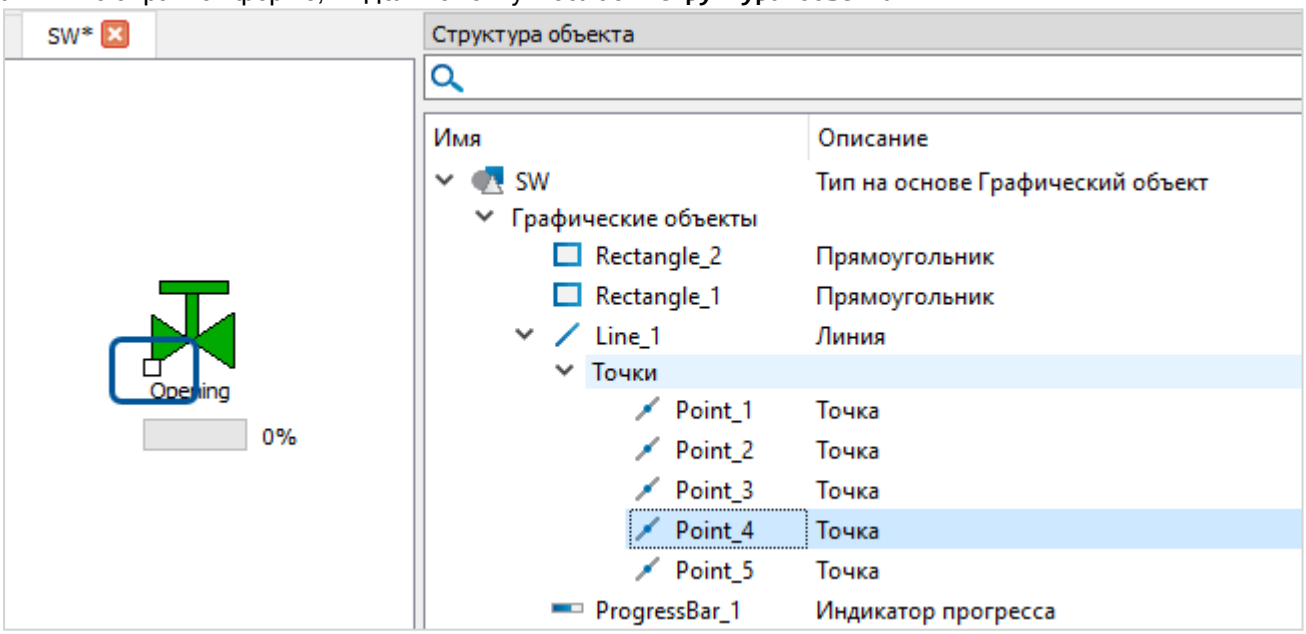
Многоугольники на мнемосхемах применяются при рисовании объектов технологического процесса. Многоугольники создаются с помощью компонента **Линия**.

Замыкание многоугольника производится путем совмещения первой и последней точки или команды **Замкнуть многоугольник** контекстного меню. Размыкание многоугольника производится командой **Разомкнуть многоугольник** контекстного меню. При размыкании удаляется линия между первой и последней точкой многоугольника.

Редактирование многоугольника возможно при его выделении и нажатии клавиши **«F2»** или командой **Редактировать многоугольник** контекстного меню. В режиме редактирования каждая точка многоугольника отмечена и доступна для перемещения и удаления. Удаление точки производится щелчком левой кнопки мыши по выбранной точке при нажатой клавише Ctrl. Добавление точки производится также в режиме редактирования при нажатой клавише **«Ctrl»**. Выход из режима редактирования производится щелчком правой кнопки мыши.

На рисунке ниже показан многоугольник и его структура из 5 точек. Координаты каждой точки определяются свойствами X и Y.

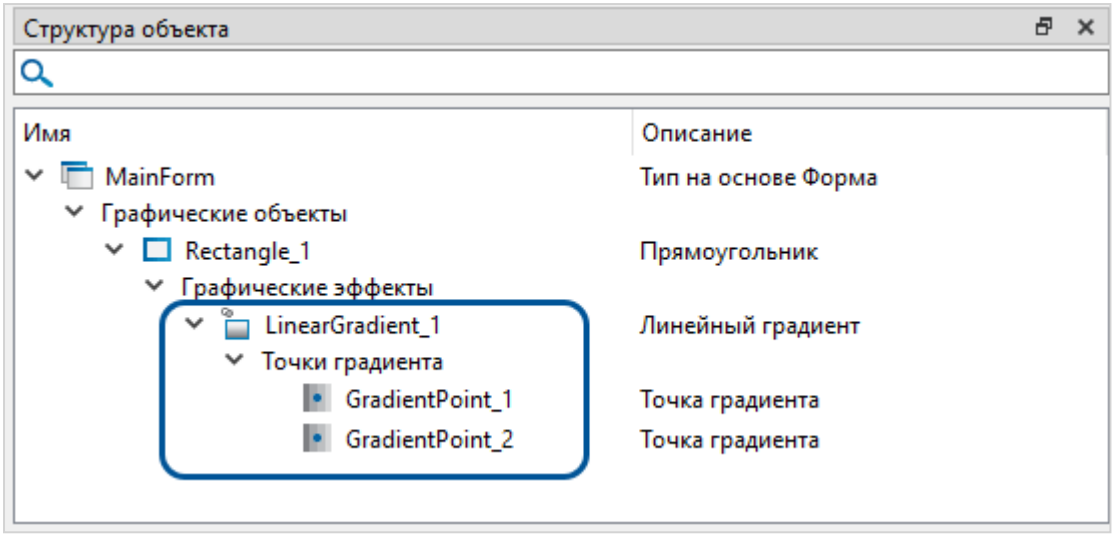
Чтобы визуально определить месторасположение конкретной точки в составе многоугольника или ломанной линии на экранной форме, выделите точку в области **Структура объекта**.



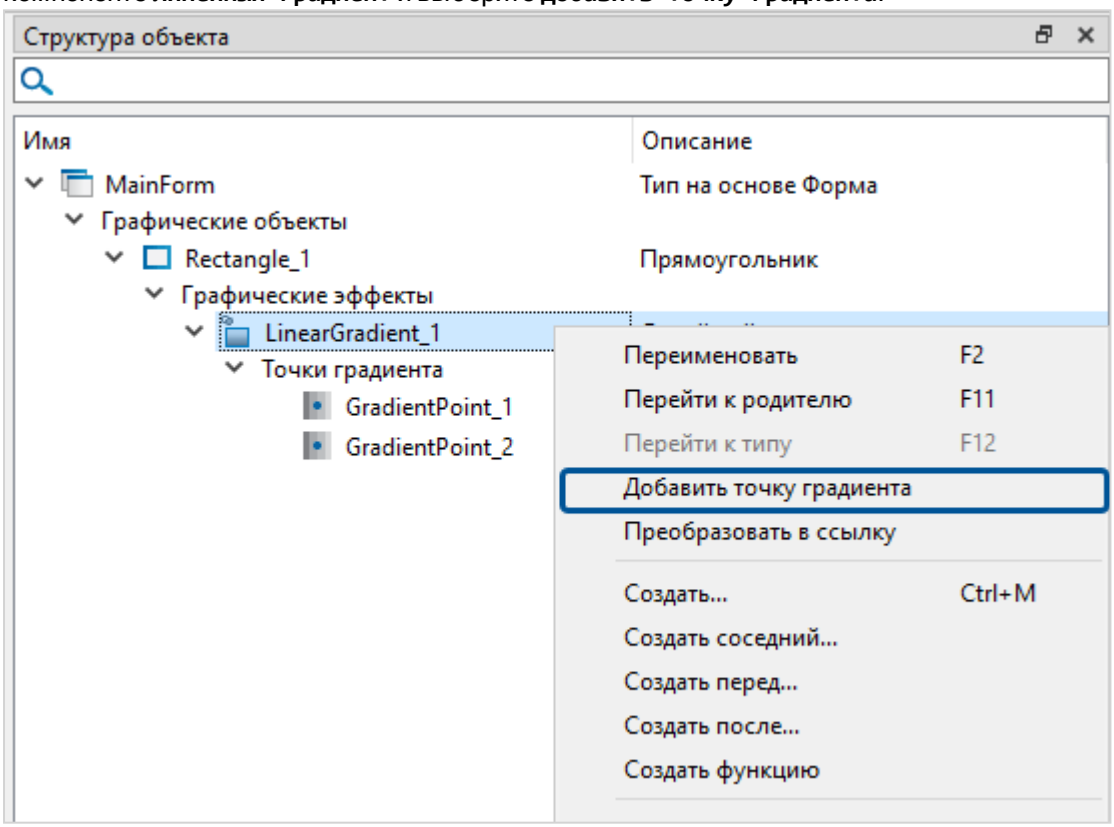
4.2.6. Линейный градиент

Линейный градиент – компонент, позволяющий создавать плавные переходы между двумя и более цветами вдоль линейной оси на графических объектах. Чтобы выполнить заливку графического объекта в виде плавного линейного перехода между двумя и более цветами, добавьте компонент **Линейный градиент** на графический объект.

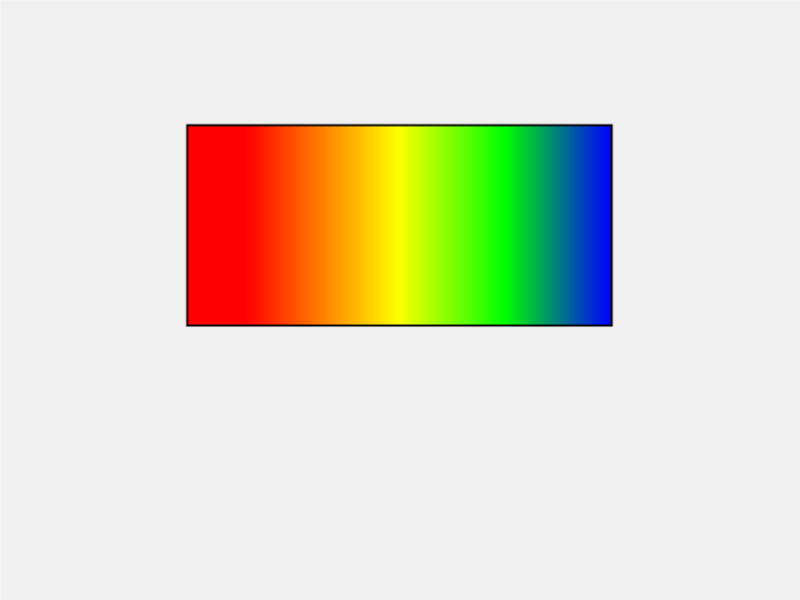
После применения линейного градиента к графическому объекту, в структуре объекта появляются точки градиента – каждая точка определяет параметры цветового перехода. Первая точка определяет начальный цвет, а последующие точки градиента указывают на другие цвета вдоль этого перехода.



Для добавления дополнительных точек градиента кликните правой кнопкой мыши на уже добавленном компоненте **Линейный градиент** и выберите **Добавить точку градиента**.



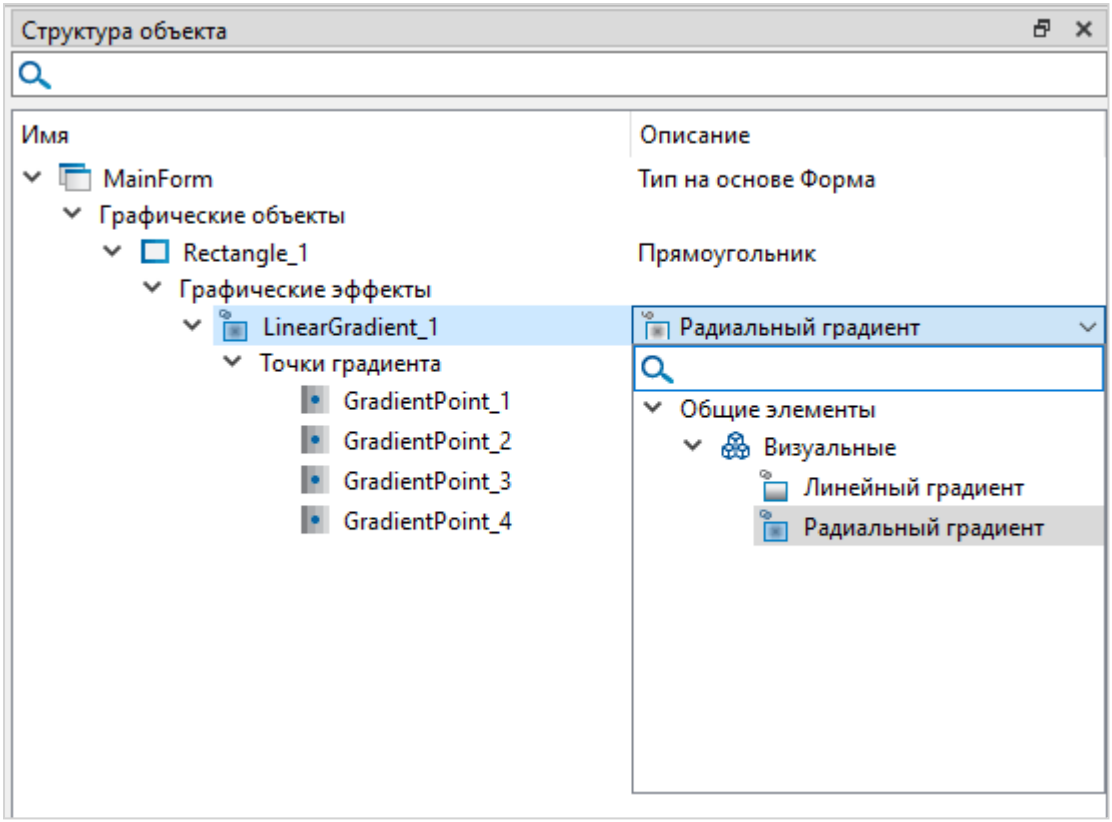
На рисунке ниже изображен графический объект, где применены четыре точки градиента, образующие линейный цветовой переход.



- Цвет в точках градиента определяется свойством **Цвет**;
- Положение каждой точки градиента определяется свойством **Положение**.
- Угол наклона градиентной линии определяется свойством **Угол поворота**.

Изменение типа градиента происходит через структуру объекта. Возможно выбрать один из двух основных типов градиента: **Линейный** и **Радиальный**. Изменение типа градиента доступно только между этими двумя видами и невозможно переключить объект с градиента на не-градиентный эффект.

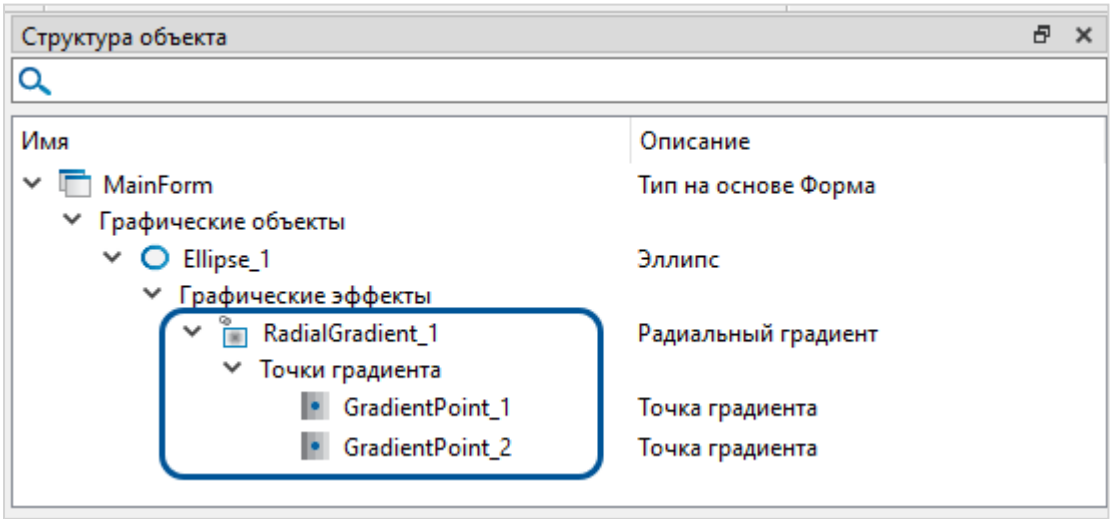
Чтобы изменить тип, в структуре выберите объект и нажмите на тип. После этого отобразится выпадающий список, из которого можно выбрать новый тип градиента для объекта, обновив тем самым его эффект заливки.



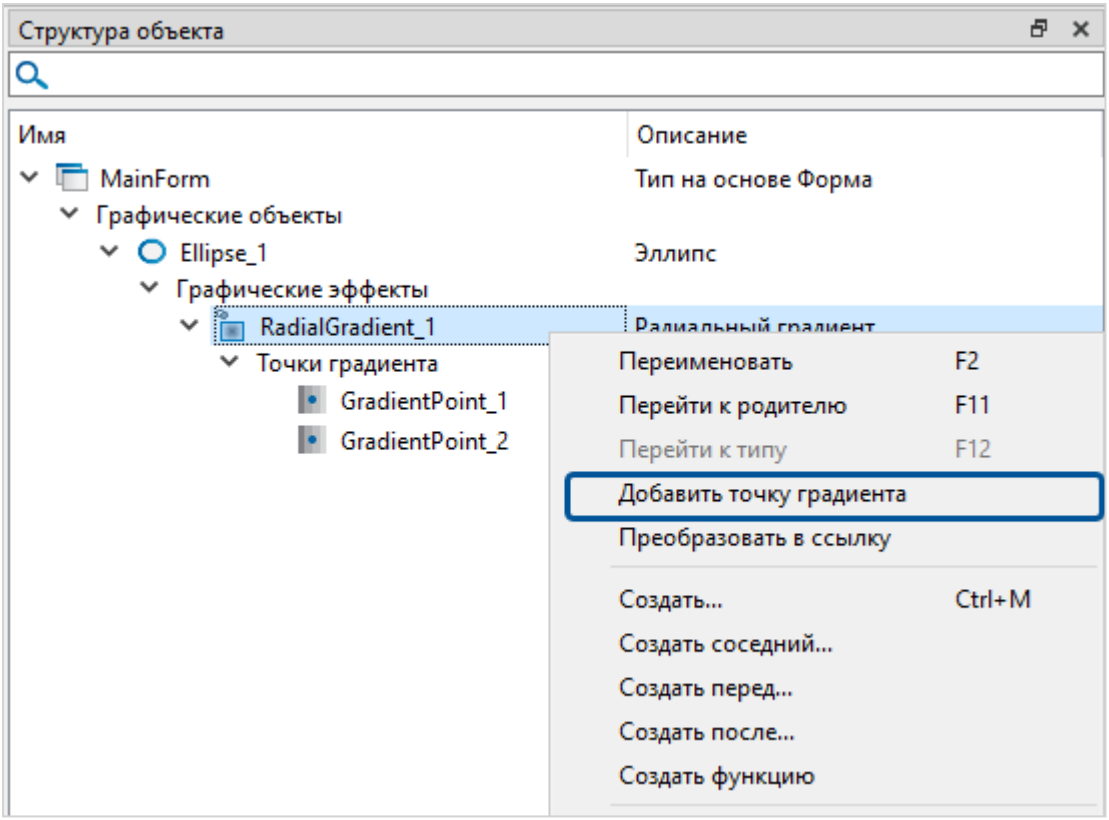
4.2.7. Радиальный градиент

Радиальный градиент – компонент, позволяющий создавать круговые переходы между двумя и более цветами. Чтобы выполнить заливку графического объекта в виде круговых переходов между двумя и более цветами, добавьте компонент **Радиальный градиент** на графический объект.

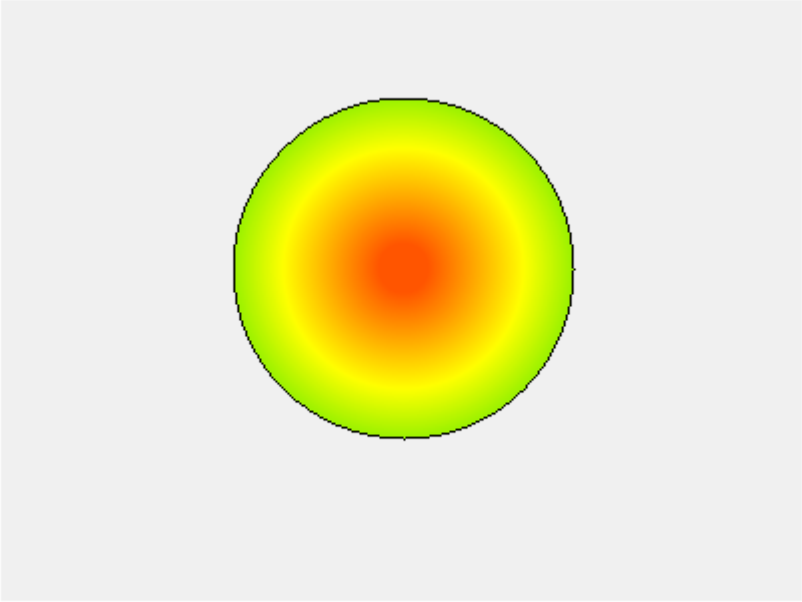
После применения радиального градиента к графическому объекту, в структуре объекта появляются точки градиента – каждая точка определяет параметры цветового перехода. Первая точка определяет начальный цвет, а последующие точки градиента указывают на другие цвета вдоль этого перехода.



Для добавления дополнительных точек градиента кликните правой кнопкой мыши на уже добавленном компоненте **Радиальный градиент** и выберите **Добавить точку градиента**.



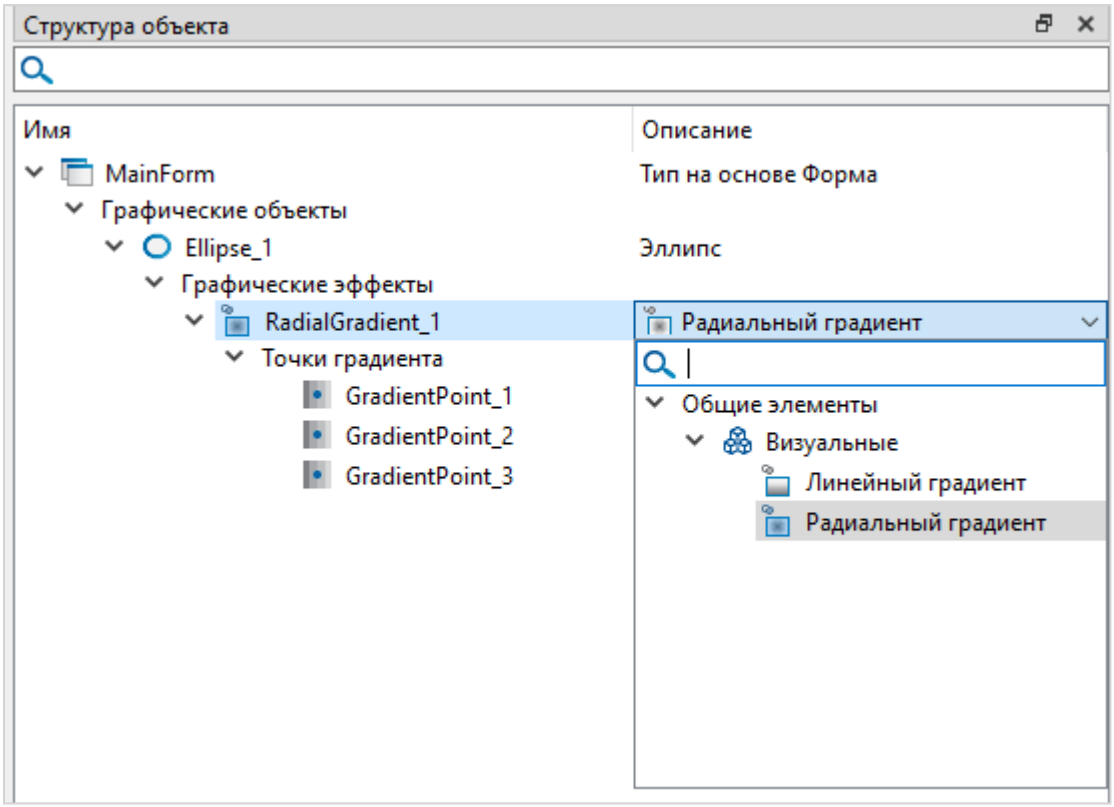
На рисунке ниже изображен графический объект, на котором используется радиальный градиент с тремя точками градиента.



- Цвет в точках градиента определяется свойством **Цвет**;
- Положение каждой точки градиента определяется свойством **Положение**.
- Координаты центра радиального градиента по горизонтали и вертикали определяются свойствами **X центра** и **Y центра**.

Изменение типа градиента происходит через структуру объекта. Возможно выбрать один из двух основных типов градиента: **Линейный** и **Радиальный**. Изменение типа градиента доступно только между этими двумя видами и невозможно переключить объект с градиента на не-градиентный эффект.

Чтобы изменить тип, в структуре выберите объект и нажмите на тип. После этого отобразится выпадающий список, из которого можно выбрать новый тип градиента для объекта, обновив тем самым его эффект заливки.



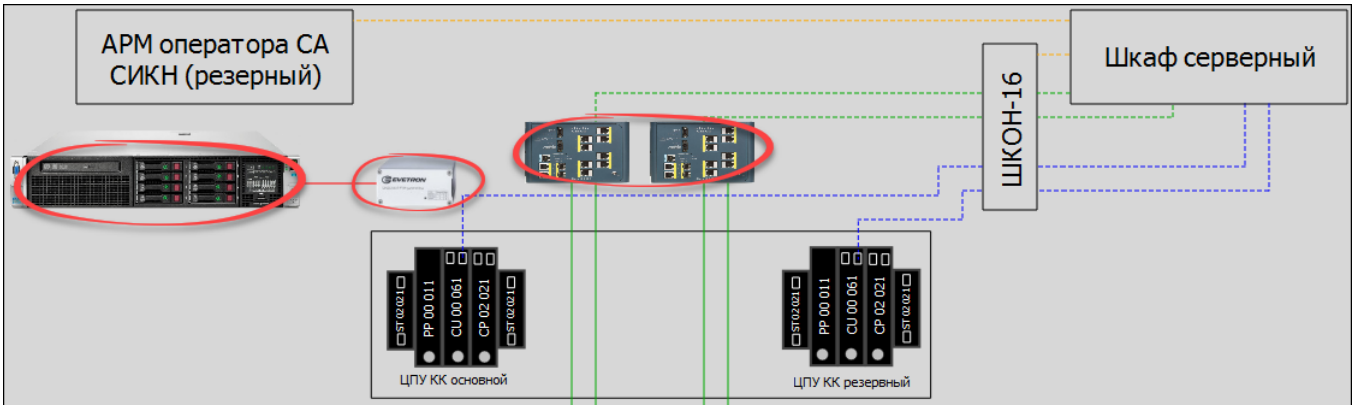
4.2.8. Изображение и анимация

Для добавления на мнемосхему растрового изображения или анимации из файла используйте компонент **Изображение**. Изображения используются для вставки логотипов компаний и визуального представления используемого оборудования. Также изображения можно использовать для воспроизведения анимации (gif-файлов). Допустимые форматы: `dds`, `gif`, `icns`, `ico`, `jpeg`, `svg`, `tga`, `tiff`, `bmp` и `webp`.

Чтобы добавить изображение, поместите его в папку проекта `resources` и укажите имя файла в свойстве **Файл изображения**. Для управления анимацией используйте свойства **Активность**, **Номер кадра** и **Скорость**.

На мнемосхемах можно применять картинки с поддержкой прозрачности.

На рисунке ниже показан участок мнемосхемы, где для иллюстрации оборудования используются растровые изображения.



4.3. Использование компонентов пользовательского интерфейса

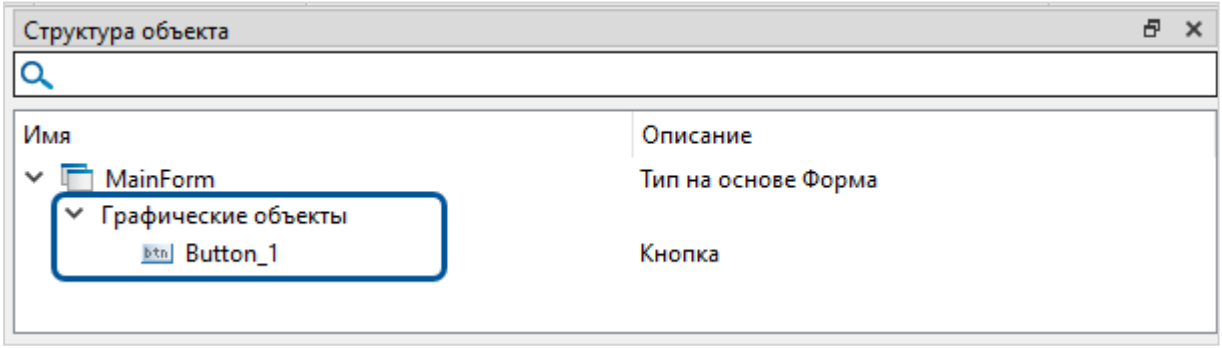
Компоненты пользовательского интерфейса (кнопки, флаги, прогресс-бары, поля ввода и т.д.) используются в проекте автоматизации для организации взаимодействия пользователя с мнемосхемой.

4.3.1. Всплывающая подсказка

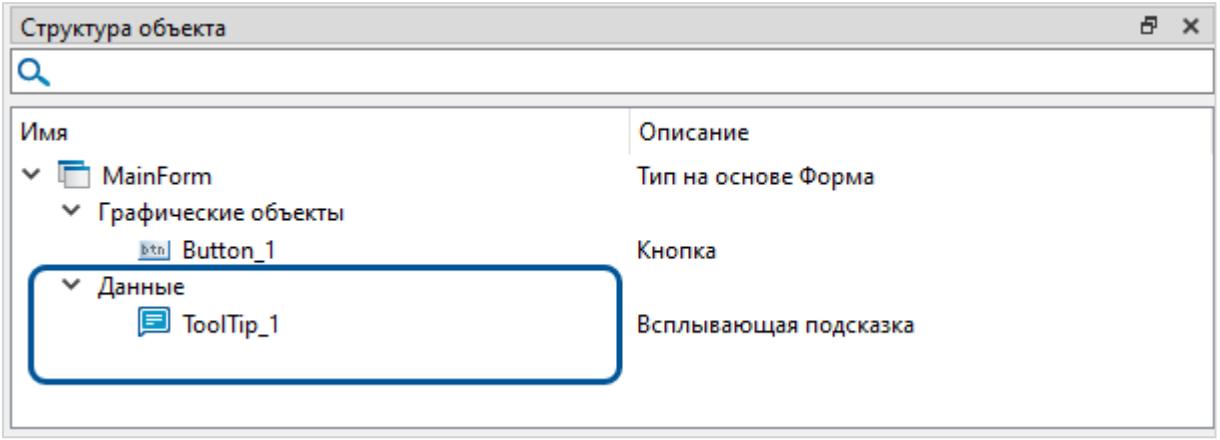
Компонент **Всплывающая подсказка** служит для отображения информативных текстовых сообщений (подсказок) в выбранном месте на экране. Этот компонент уникален тем, что его можно использовать независимо от других графических элементов, позволяя отображать подсказки в любой точке рабочего пространства.

4.3.1.1. Как настроить всплывающую подсказку для кнопки

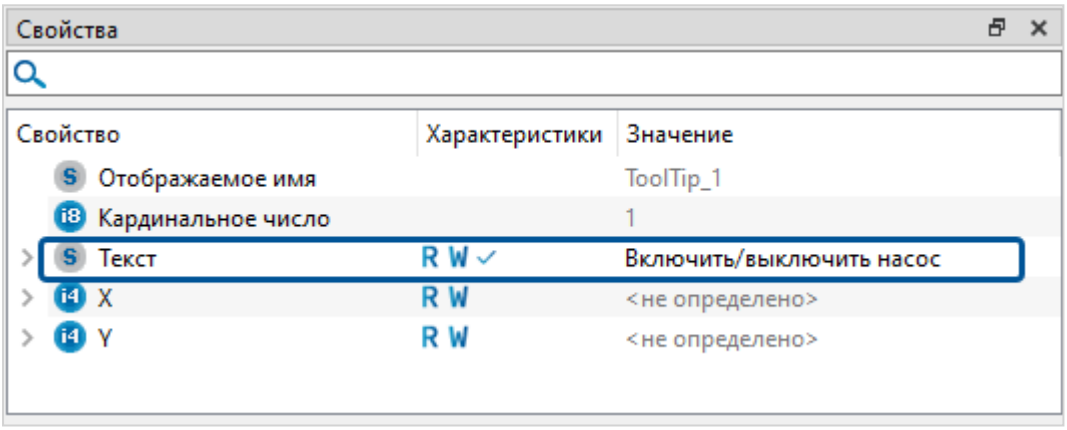
1. Добавьте на форму компонент **Кнопка**, для которой будет настраиваться всплывающая подсказка.



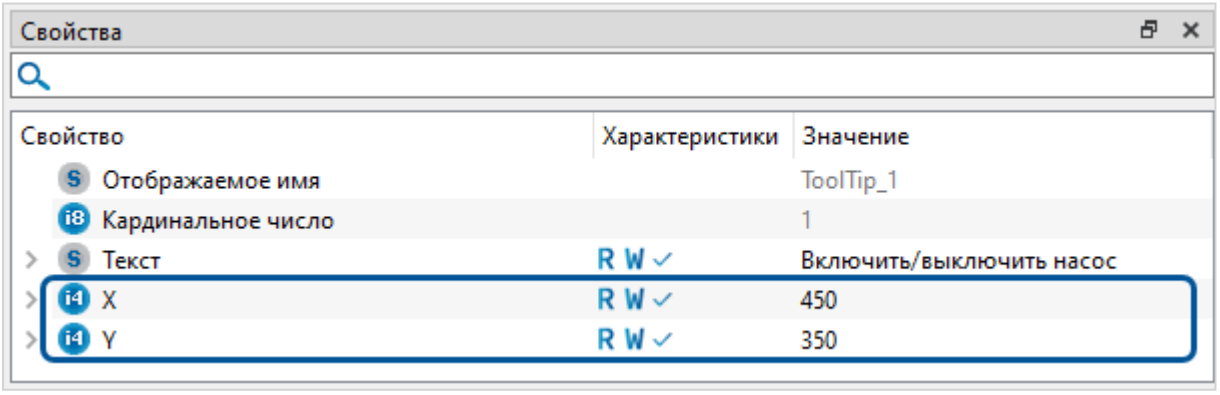
2. Также добавьте на форму компонент **Всплывающая подсказка**. Этот компонент будет использоваться для отображения текстовой информации при взаимодействии с кнопкой.



3. Задайте текст подсказки в свойстве Текст компонента **Всплывающая подсказка**.



4. Определите координаты X и Y через свойства X и Y компонента **Всплывающая подсказка** для указания местоположения подсказки на экране.



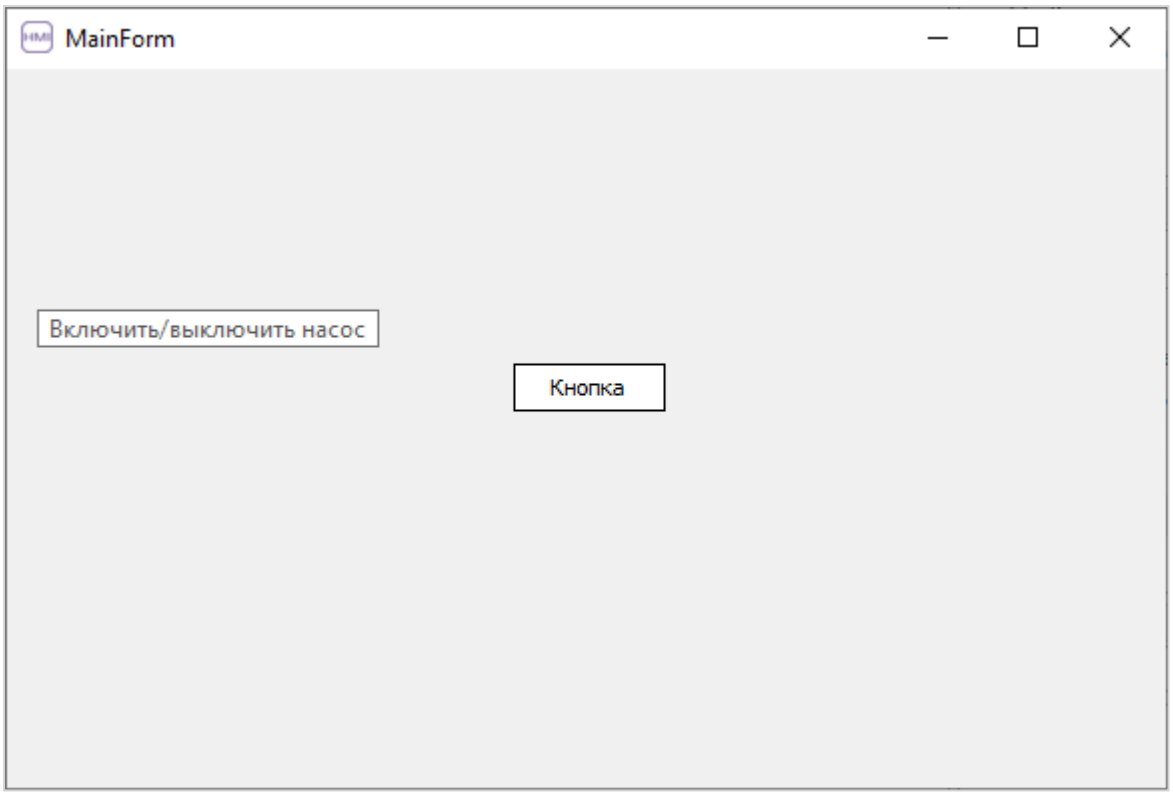
5. Настройте обработчики событий MouseEnter и MouseLeave для кнопки. В обработчике события MouseEnter используйте метод Show() для компонента **Всплывающая подсказка**, чтобы отобразить подсказку при наведении курсора на кнопку. Пример кода на языке SePlatform.Om:

```
ToolTip_1.Show();
```

В обработчике события MouseLeave используйте метод Hide() для компонента **Всплывающая подсказка**, чтобы скрыть подсказку при уходе курсора с кнопки. Пример кода на языке SePlatform.Om:

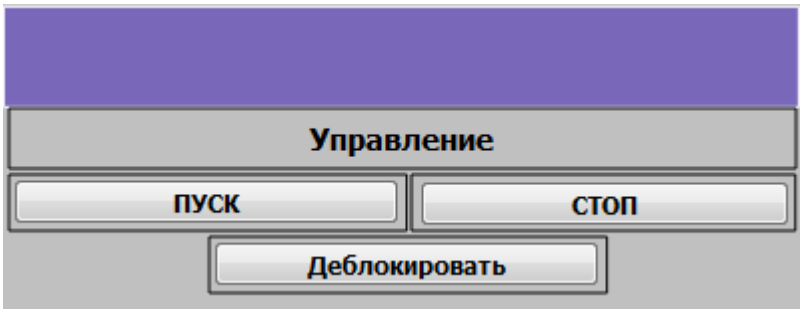
```
ToolTip_1.Hide();
```

6. Теперь, в режиме рантайма при наведении курсора на кнопку, всплывающая подсказка будет отображаться, а при уходе курсора с кнопки она будет скрыта.



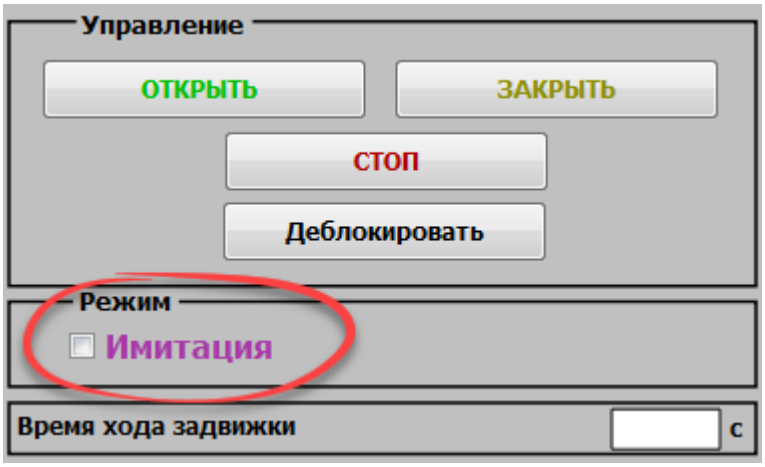
4.3.2. Кнопка

Компонент **Кнопка** применяется на мнемосхемах для подачи управляющих команд или взаимодействия с пользователем в диалоговых окнах. Действия, происходящие при нажатии на кнопку, определяются в обработчике события ButtonPressed. На рисунках ниже показана экранная форма с кнопками для подачи команд управления технологическим объектом.



4.3.3. Флажок

Компонент **Флажок** применяется на мнемосхемах для организации опциональных (включаемых или отключаемых) возможностей. На рисунке ниже показана экранная форма, где флажком активируется имитационный режим для подаваемых команд.



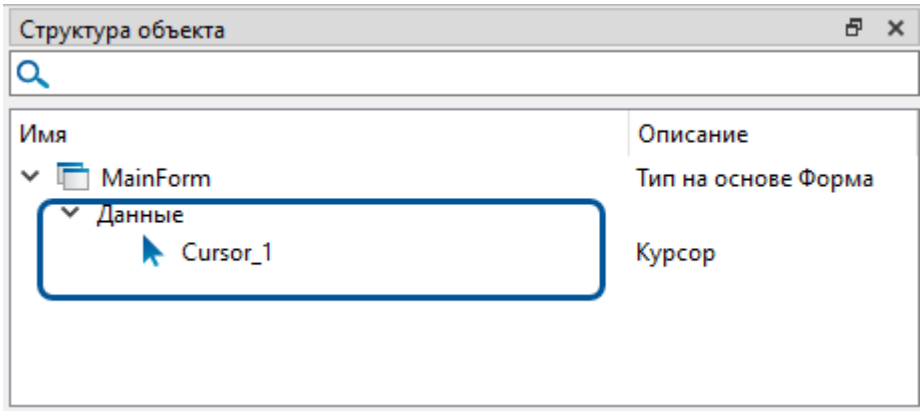
- надпись возле флажка регулируется свойством Текст;
- свойство Состояние определяет текущее положение флажка (включен/выключен);
- действия, при смене состояния флажка определяются в обработчике события StateChanged.

4.3.4. Курсор

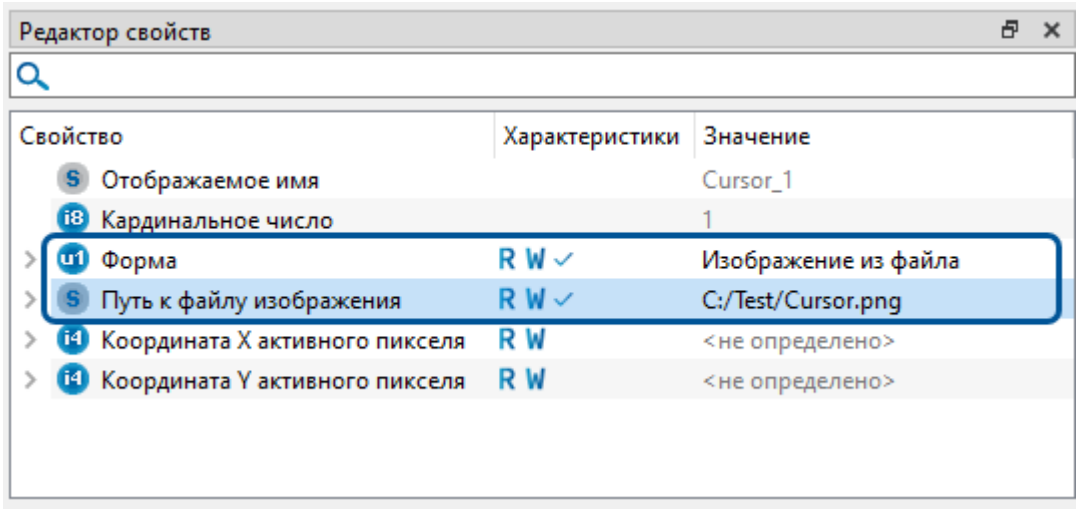
Компонент **Курсор** позволяет настраивать внешний вид курсора в проекте SePlatform.HMI. Компонент предоставляет доступ к набору стандартных форм курсора, интегрированных в операционную систему, а также дает возможность использовать собственные изображения для создания уникальных курсоров.

4.3.4.1. Как изменить вид курсора в проекте SePlatform.HMI

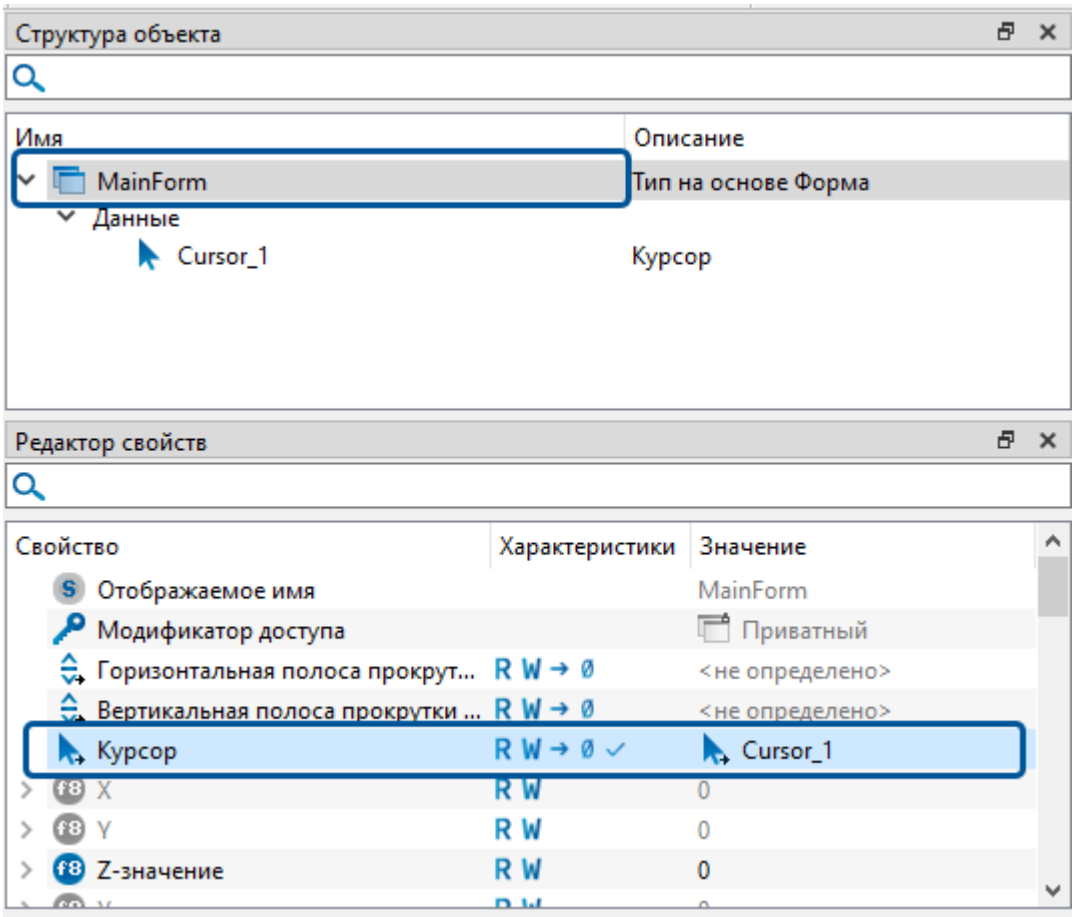
1. Чтобы изменить внешний вид курсора, добавьте на форму компонент **Курсор**.



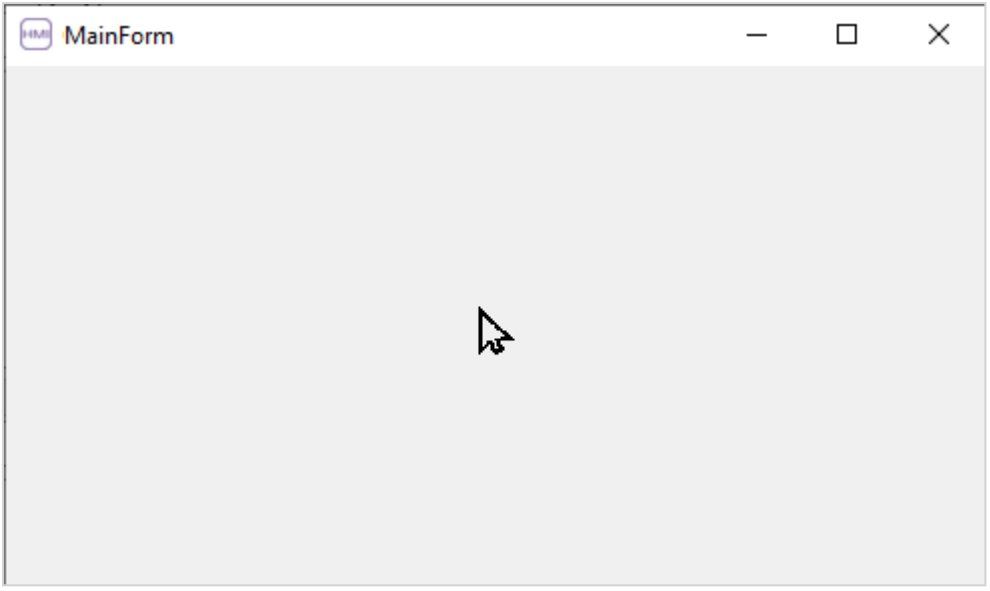
2. Для использования индивидуального изображения курсора, выберите опцию **«Изображение из файла»** в свойстве **Форма** и укажите путь к вашему изображению курсора через свойство **Путь к файлу изображения**.



3. После настройки внешнего вида курсора перейдите в свойства экранной формы и добавьте ссылку на объект **Курсор**. Для этого нажмите правой кнопкой мыши по свойству **Курсор** → **Сослаться** → **Cursor_1**.



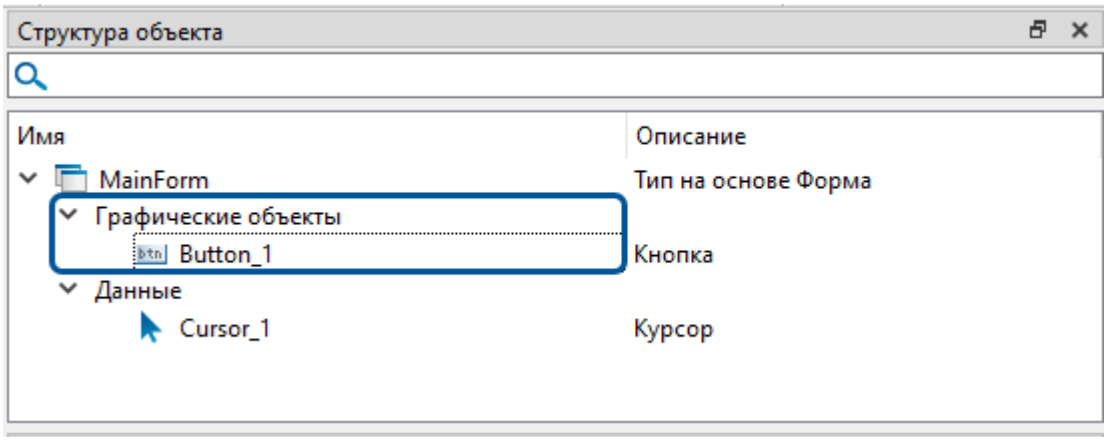
4. Теперь при открытии формы в рантайме будет отображаться новый курсор.



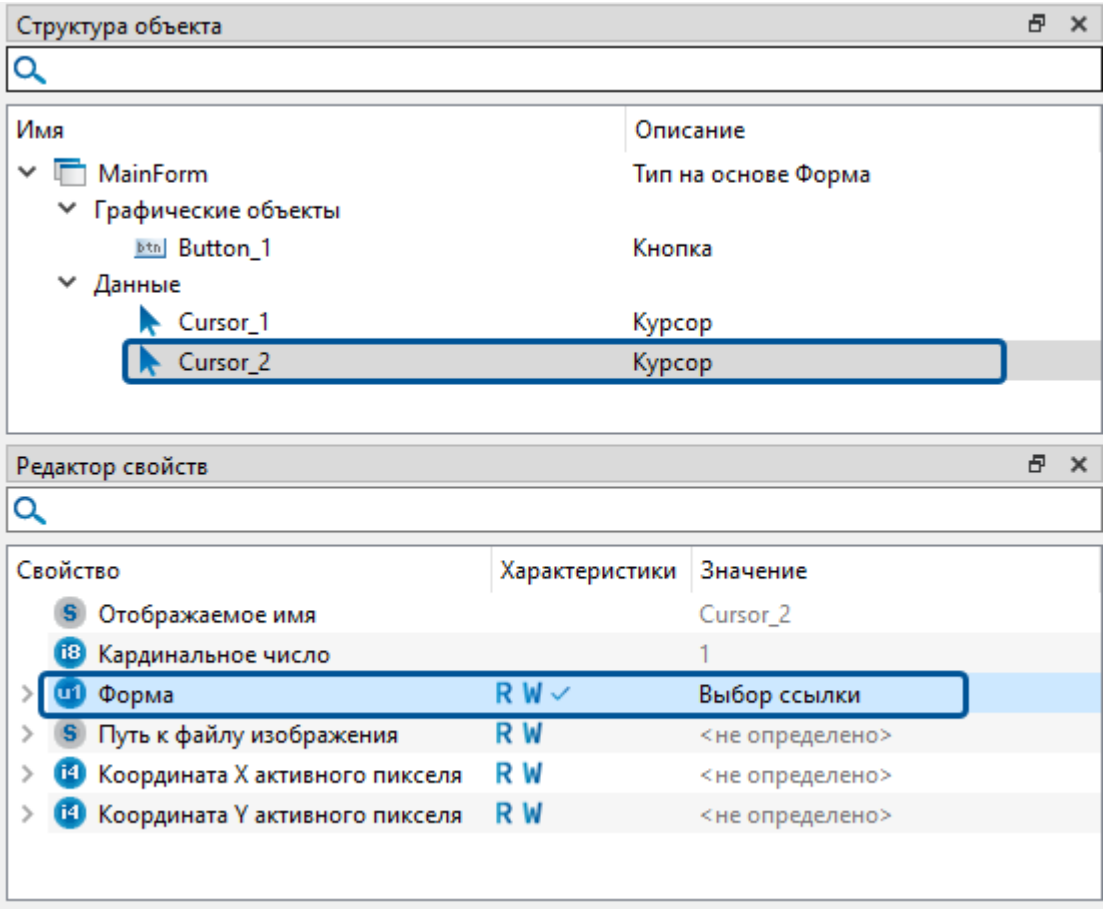
4.3.4.2. Как настроить изменение вида курсора при наведении на графический объект

Для всех графических объектов в SePlatform.HMI, которые обладают свойством **Курсор**, можно задать индивидуальный курсор. Этот курсор будет автоматически отображаться при наведении на объект.

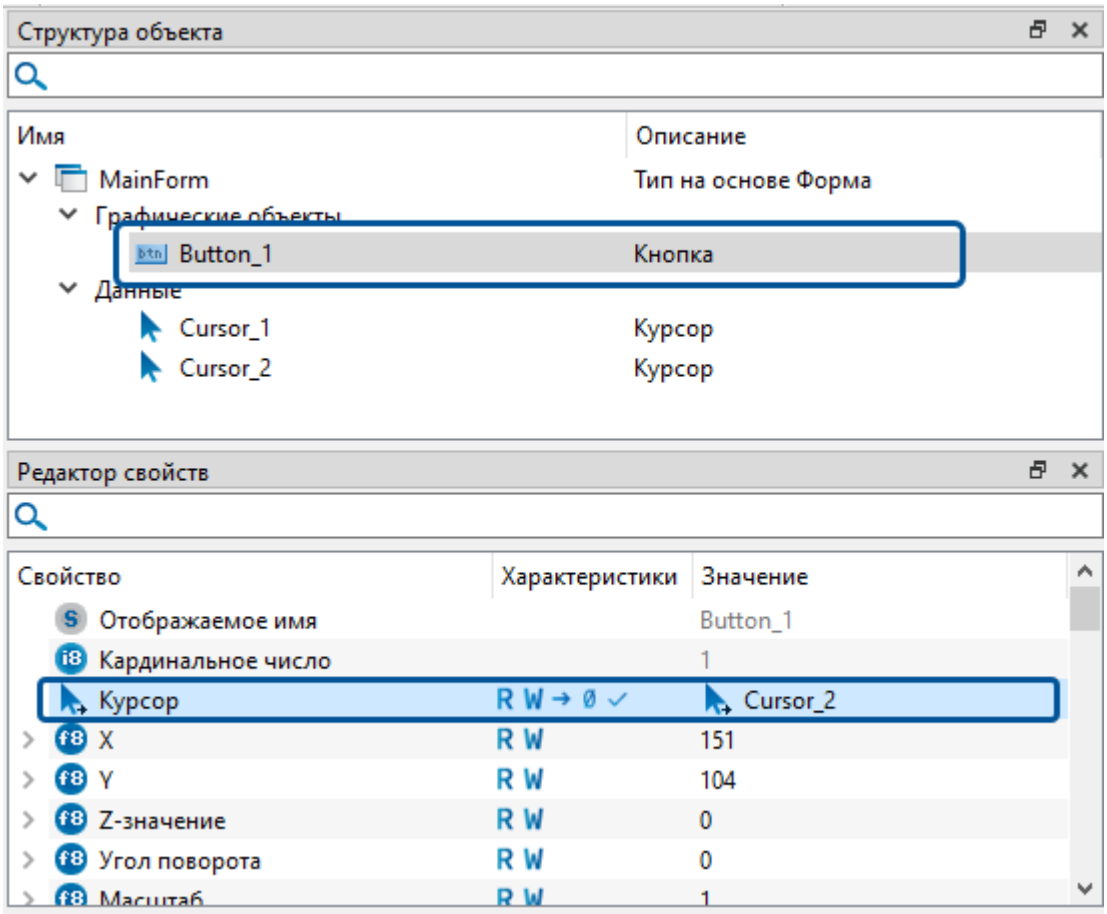
1. Добавьте на форму компонент **Кнопка**.



2. Чтобы при наведении курсора на **Кнопку** отображался курсор в виде ладошки с пальцем, добавьте еще один объект **Курсор**. В свойстве **Форма** выберите опцию **«Выбор ссылки»**.

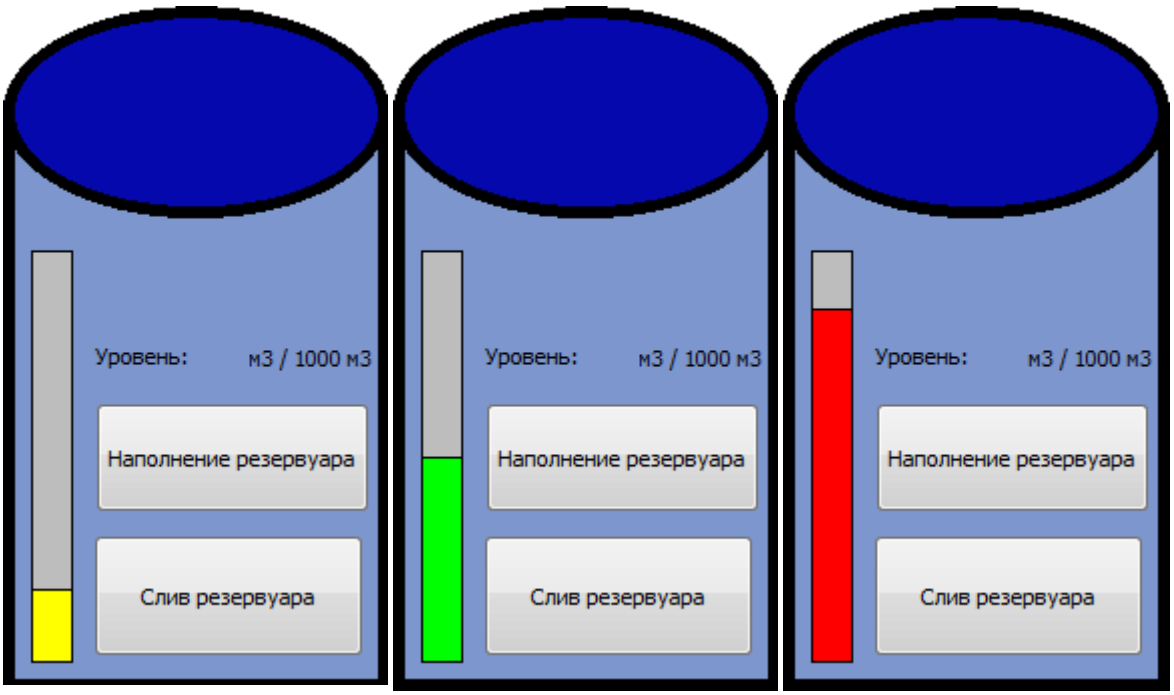


3. В редакторе свойств у объекта **Кнопка** выберите свойство Курсор. Установите ссылку на ранее добавленный объект **Курсор**. Для этого нажмите правой кнопкой мыши по свойству Курсор → **Сослаться** → **Cursor_2**.



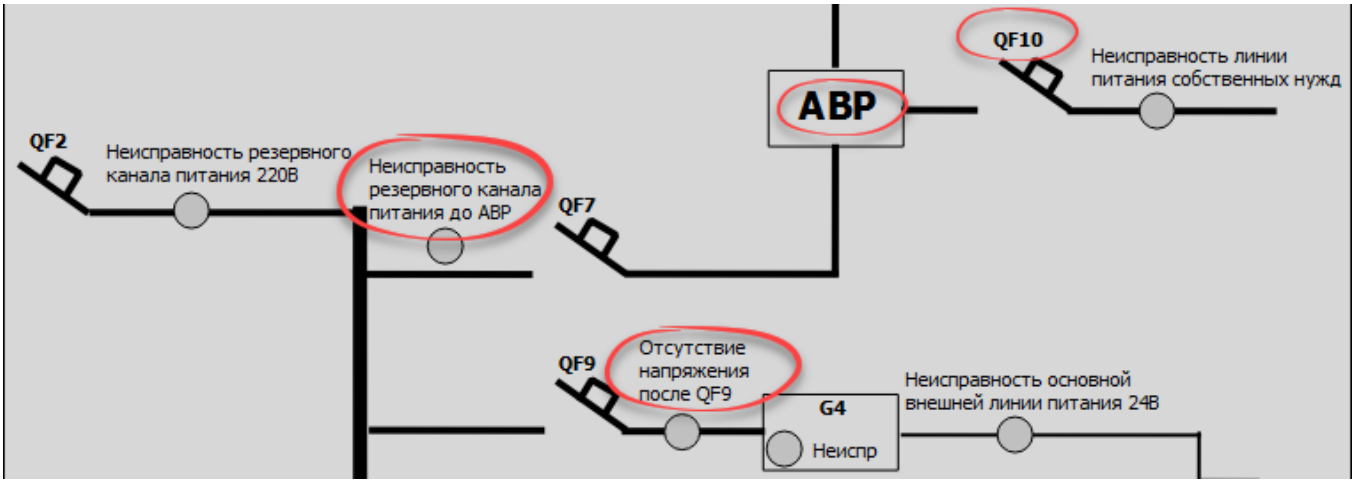
4.3.5. Индикатор гистограммы

Компонент **Индикатор гистограммы** применяется на мнемосхемах для демонстрации значений некоторых динамических параметров, у которых есть нижнее и верхнее пороговое значение. Текущее значение индикатора гистограммы определяется свойством Значение. На рисунках ниже гистограмма применена для отображения уровня продукта в резервуаре. На первом рисунке уровень продукта ниже Нижнего порогового значения (индикатор - желтый), на втором - уровень продукта в пределах нормы (зеленый), на третьем - уровень превысил Верхнее пороговое значение (красный).



4.3.6. Текст

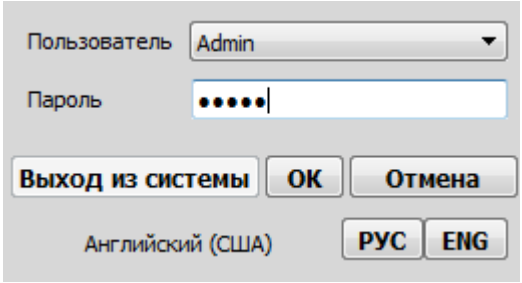
Компонент **Текст** применяется на мнемосхемах для пояснительных целей. На рисунке ниже показан участок мнемосхемы, где используется множество текстовых блоков для пояснения или именования элементов.



- содержимое текстового блока содержится в свойстве Текст;
- стиль, размер и цвет шрифта регулируется свойствами Шрифт и Цвет шрифта;
- положение текста внутри блока регулируется свойством Выравнивание текста.

4.3.7. Поле ввода

Компонент **Поле ввода** применяется на мнемосхемах в местах, где требуется ручной ввод неких значений. На рисунке ниже показана экранная форма с полем ввода для пароля.



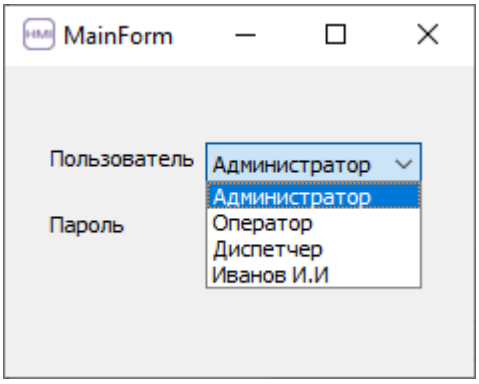
- содержимое поля ввода по умолчанию регулируется свойством Текст;
- режим маскирования содержимого звездочками (для ввода пароля) включается в свойстве Скрывать ввод;
- действия, происходящие по завершении редактирования поля ввода , определяются в обработчике события EditFinished.

4.3.8. Выпадающий список

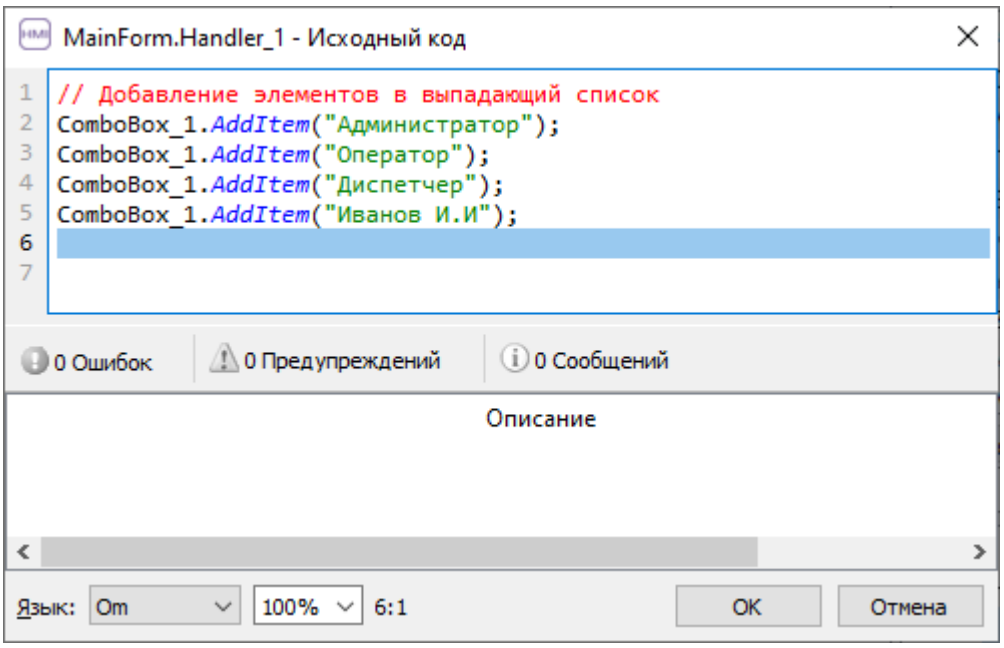
Компонент **Выпадающий список** предоставляет возможность выбора одного из нескольких заранее определенных значений.

Выпадающий список используется для различных сценариев, например выбор пользователей для авторизации в системе безопасности или других параметров, где требуется выбор из заранее определенных вариантов.

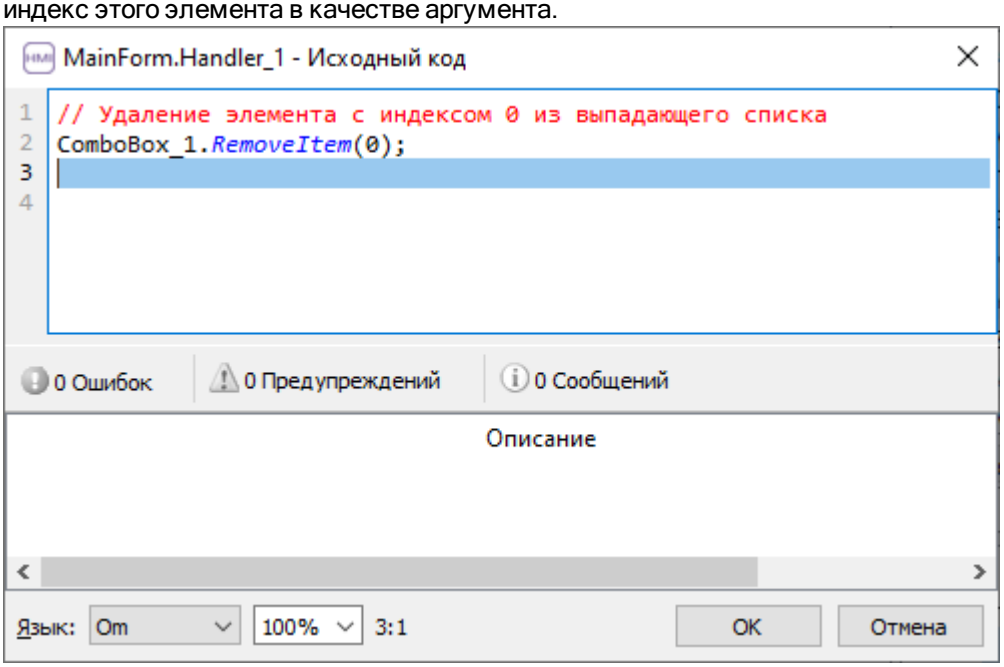
На рисунке ниже показана экранная форма с выпадающим списком пользователей для авторизации в подсистеме безопасности.



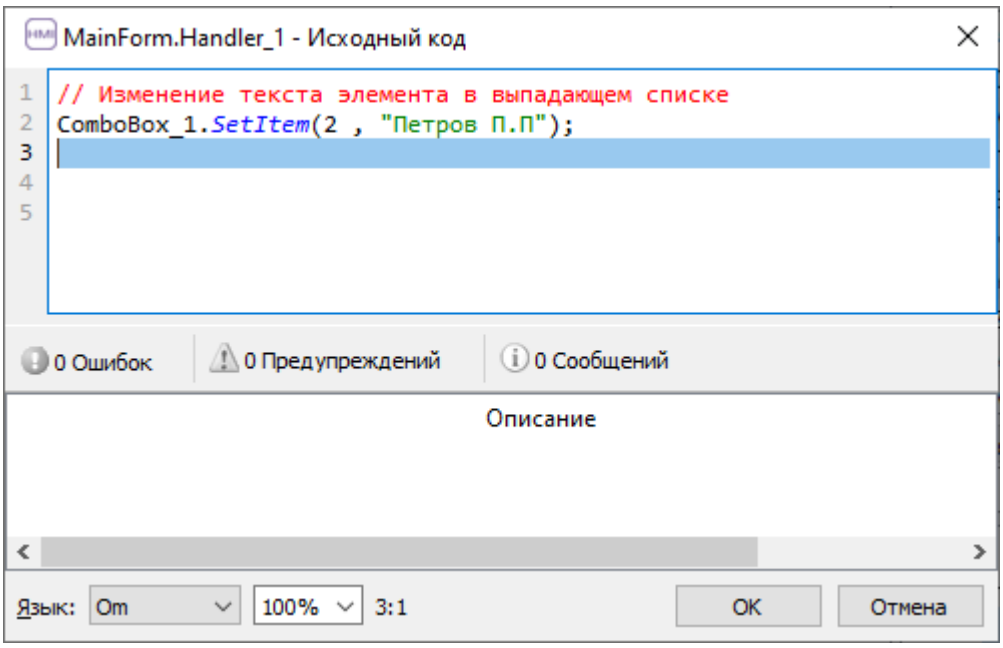
Чтобы заполнить элементами выпадающий список, используйте функцию AddItem().



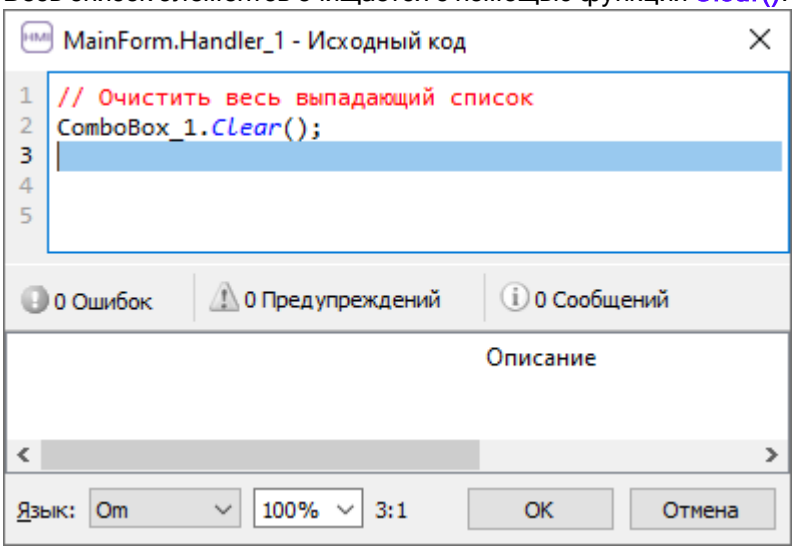
Каждый элемент в списке имеет свой индекс. Нумерация элементов списка начинается с 0.
Чтобы удалить ненужный элемент из выпадающего списка, используйте функцию RemoveItem(), указав индекс этого элемента в качестве аргумента.



Чтобы обновить текст конкретного элемента используйте функцию SetItem(), указав индекс этого элемента и новый текст, который необходимо установить.

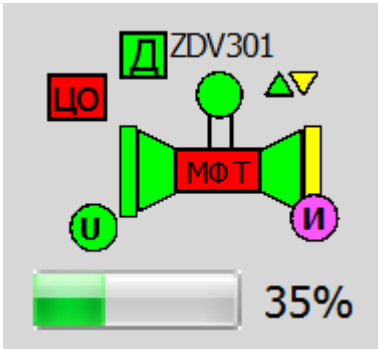


Весь список элементов очищается с помощью функции Clear().



4.3.9. Индикатор прогресса

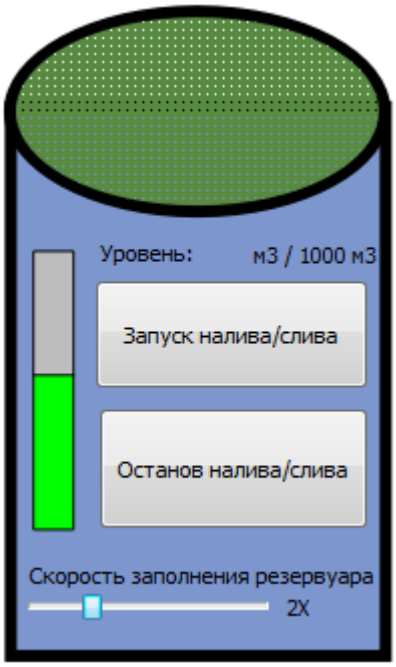
Компонент **Индикатор прогресса** используется для отображения процентной доли завершенности процесса. На рисунке ниже показан индикатор прогресса, который отражает процесс открытия задвижки.



- свойство **Значение** содержит текущий показатель индикатора прогресса;
- режим ориентации индикатора (горизонтальный или вертикальный) устанавливается в свойстве **Ориентация**.

4.3.10. Ползунок

Компонент **Ползунок** применяется для выставления некоторых фиксированных значений для параметров. Ниже показан элемент мнемосхемы - резервуар, скорость его заполнения оператор может регулировать ползунком, выставляя значения от 1X до 5X.



- свойство **Значение** содержит текущее значение ползунка;
- режим ориентации ползунка (горизонтальный или вертикальный) устанавливается в свойстве **Ориентация**.
- действия, происходящие при изменении положения ползунка, определяются в обработчике события **ButtonPressed**.

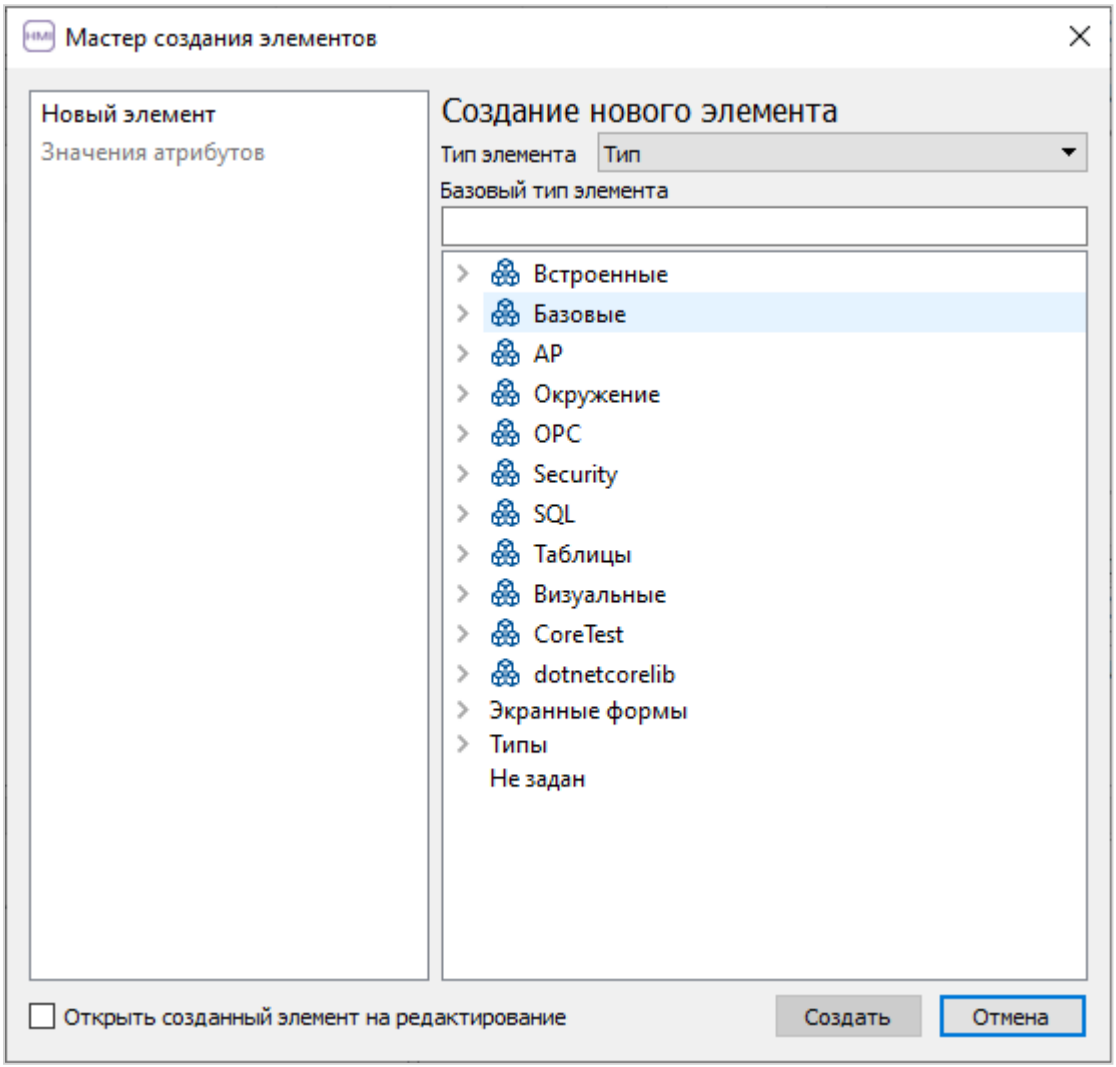
5. Типизация и объектный подход

5.1. Создание типов

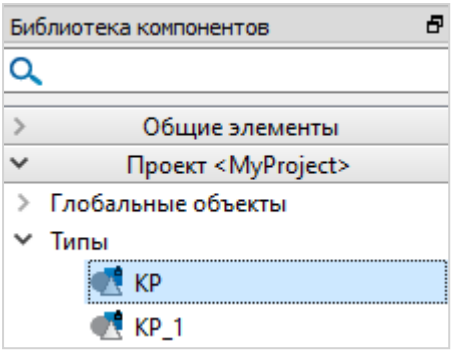
5.1.1. Создание типов графических объектов

Для многократного применения однотипных объектов на мнемосхемах создаются типы графических объектов. Разработанный единожды тип графического объекта может многократно применяться на разных экранных формах проекта автоматизации.

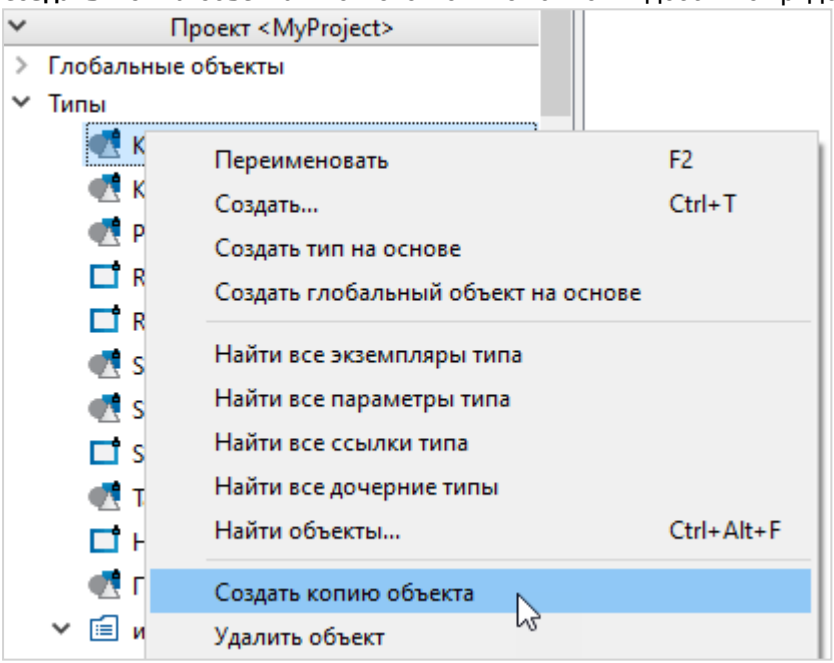
Чтобы создать новый тип, выберите команду контекстного меню **Создать** и воспользуйтесь мастером создания элементов.



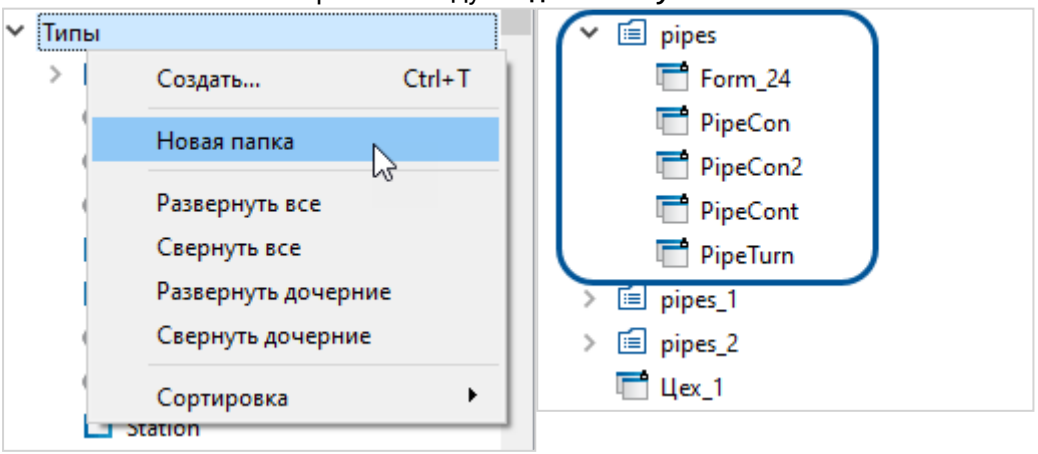
Создастся новый тип со стандартным названием. Прототип, на основе которого создан тип, отображается во всплывающей подсказке. Для редактирования типа дважды щелкните по нему.



При первичном редактировании типа появится чистая рабочая область. Возможности работы в рабочей области типа аналогичны возможностям работы в рабочей области экранной формы.
Созданные типы сохраняются в папке проекта `objects` с расширением файлов `omobj`.
Чтобы создать тип на основе компонентов стандартной библиотеки компонентов, выберите команду контекстного меню **Создать тип графического объекта** → **На основе** → **Общие элементы** → **[Название компонента]**.
Чтобы скопировать тип графического объекта, в **Библиотеке компонентов** выберите тип и выполните команду **Создать копию объекта** в контекстном меню. Копия добавится рядом с копируемым типом.



Чтобы группировать типы графических объектов, используйте папки. Для создания папки перейдите в контекстное меню и выберите команду **Создать папку**.



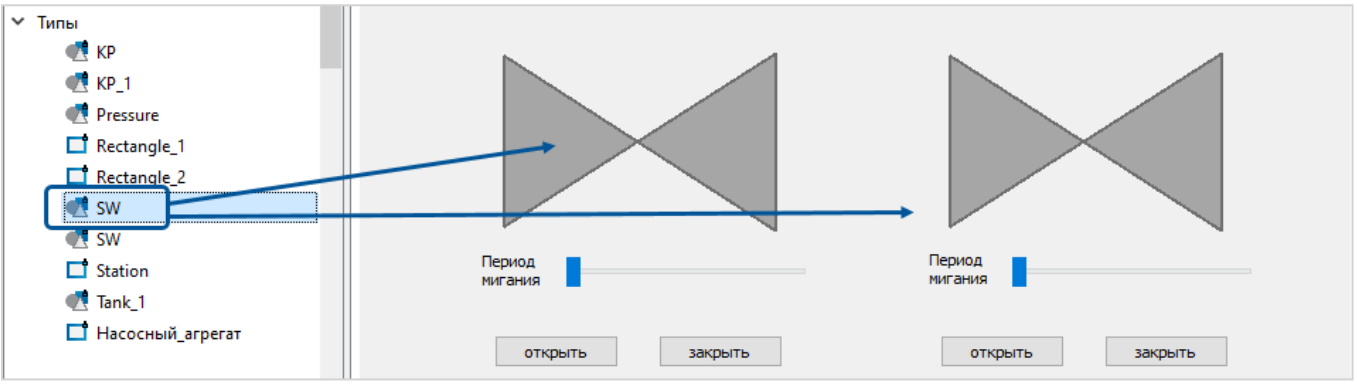
ОБРАТИТЕ ВНИМАНИЕ

Нельзя допускать совпадения имён объектов, находящихся в папках и вне папок. Например, если имя объекта в папке совпадёт с именем объекта вне папки, то ссылка будет произведена на объект размещённый вне папки.

ОБРАТИТЕ ВНИМАНИЕ

Чтобы обратиться к графическому объекту, расположенному в папке, в коде указывайте только имя объекта без имени папки.

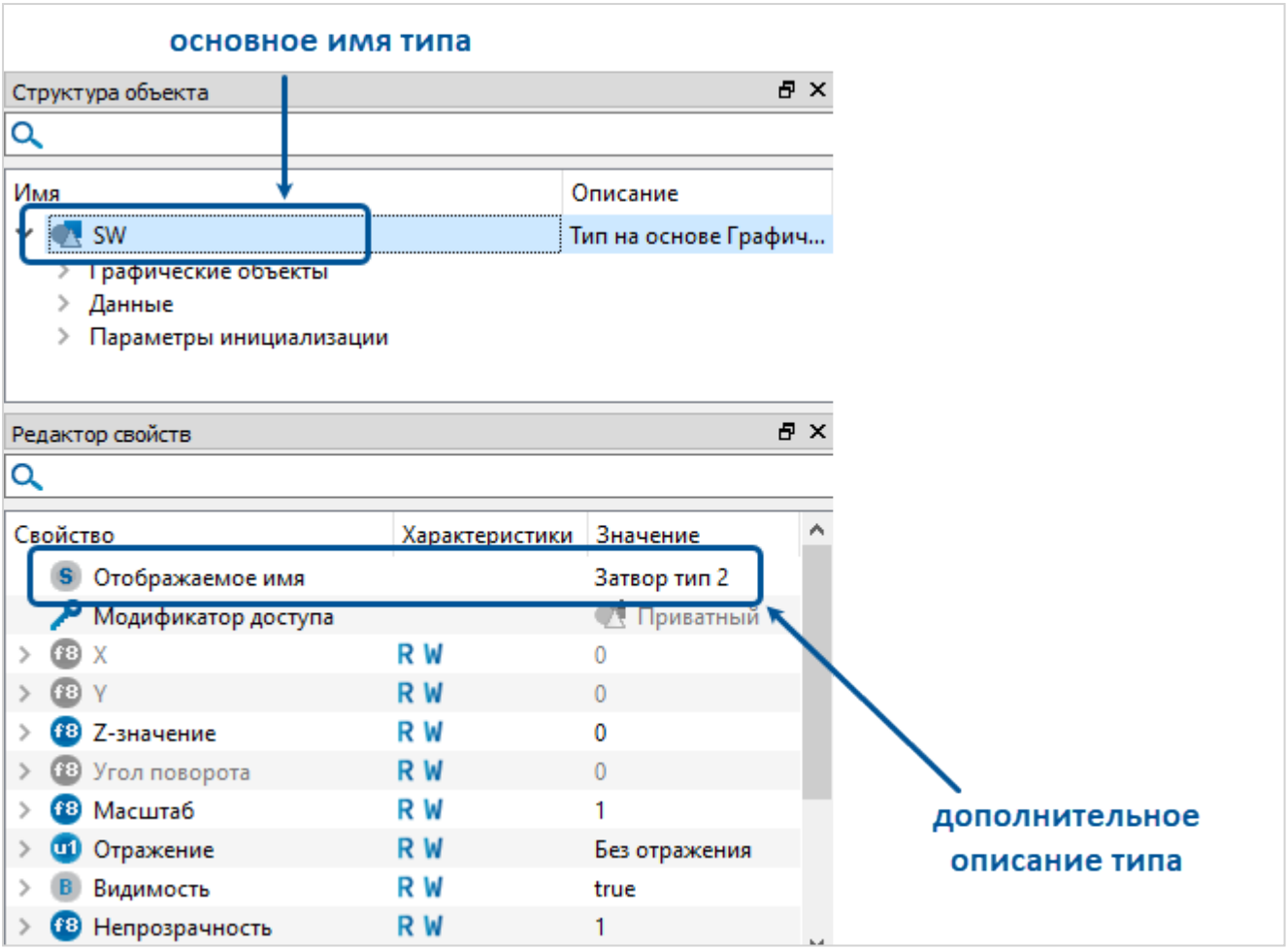
Чтобы добавить экземпляр объекта созданного типа, перетащите нужный тип на рабочую область экранной формы или рабочую область другого типа.



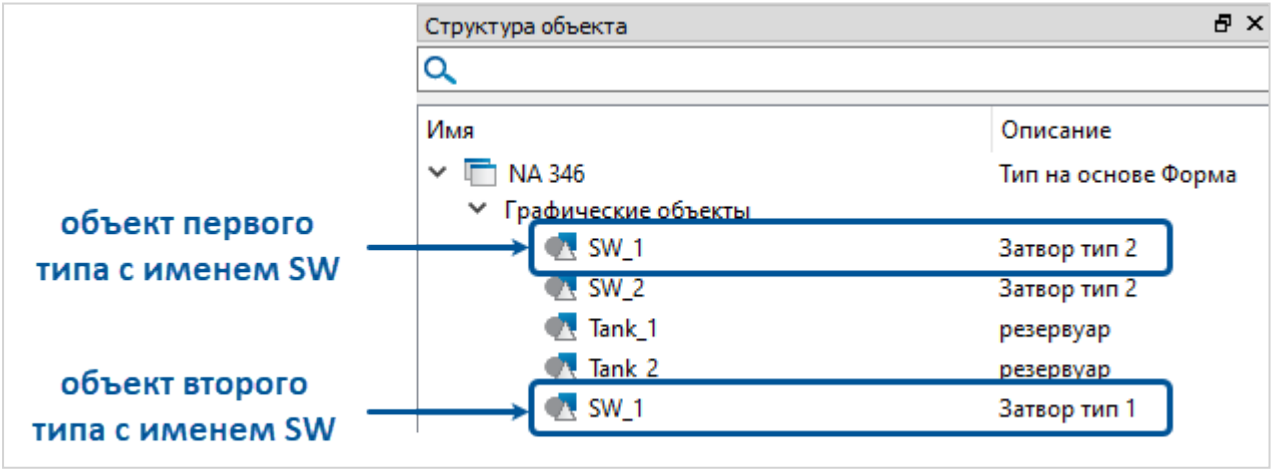
Чтобы перейти к редактированию типа графического объекта, находящегося на форме, выберите объект и нажмите «**F12**» или в контекстном меню объекта выберите **Перейти к типу**. В результате тип графического объекта будет открыт для редактирования и выделен в **Библиотеке компонентов**.

5.1.2. Добавление описаний

Чтобы добавить произвольное дополнительное описание каждому созданному типу, используйте свойство **Отображаемое имя**.

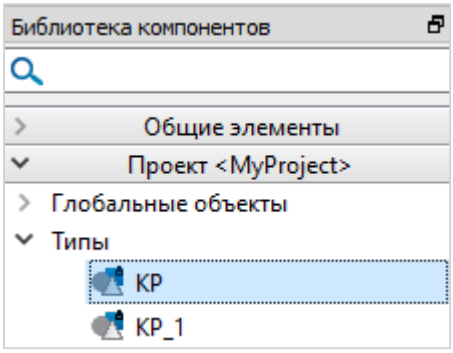


После добавления экземпляра (объекта) типа на форму, описание будет отображаться рядом с именем экземпляра в области **Структура объекта**. Благодаря наличию отображаемого имени можно ориентироваться среди объектов с одинаковыми именами, но разными типами.



5.1.3. Редактирование типов графических объектов

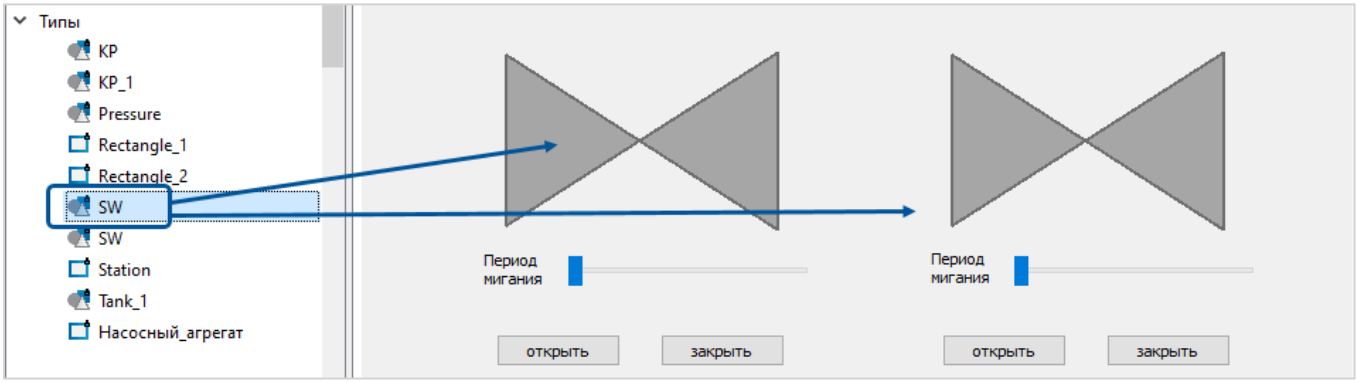
Для редактирования типа дважды щелкните по нему.



При первом редактировании типа появится чистая рабочая область. Действия в рабочей области типа аналогичны действиям в рабочей области экранной формы (стр. 1).
Чтобы перейти к редактированию типа конкретного экземпляра объекта данного типа выделите экземпляр и в контекстном меню выберите **Перейти к типу** (клавиша **«F12»**). В результате тип графического объекта будет открыт для редактирования и выделен в **Библиотеке компонентов**.

5.1.4. Добавление в проект экземпляров типа

Чтобы добавить на форму экземпляр созданного типа, перетащите нужный тип на рабочую область экранной формы или рабочую область другого типа.



Для каждого добавленного экземпляра вы можете задать свои значения свойств в **Редакторе свойств**.

5.1.4.1. Работа с параметрами инициализации

Экземпляры объектов графических типов и экранные формы как правило имеют параметры инициализации, которые нужны для определения специфичных параметров каждого отдельно взятого экземпляра.

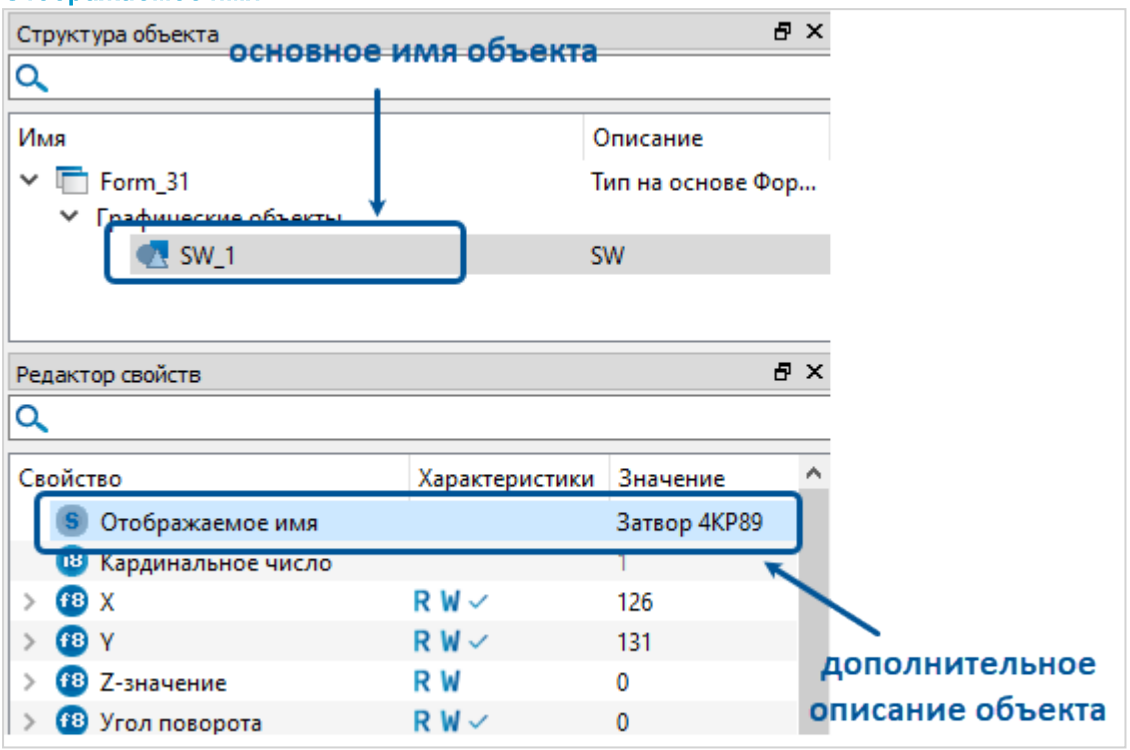
Параметрами инициализации объектов могут быть:

- Значения примитивных типов (**Создать** → **Параметр инициализации** → **Примитивный тип** → **<Имя типа>**);
- Ссылки на объекты стандартной библиотеки компонентов (**Создать** → **Параметр инициализации** → **Ссылка** → **На основе** → **Общие элементы** → **<Имя элемента>**);
- Ссылки на объекты графических типов.

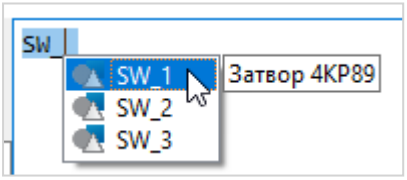
Параметры инициализации объектов позволяют указывать начальные значения свойствам (стр. 69). После запуска экранной формы в рантайме свойства объектов, для которых было указано инициализирующее значение, примут данные значения.

5.1.4.2. Добавление описаний

Чтобы добавить произвольное дополнительное описание экземпляру типа, используйте свойство **Отображаемое имя**.



Благодаря наличию отображаемого имени можно обращаться к объектам в коде по англоязычному имени, ориентируясь среди объектов при помощи подсказок с отображаемым именем.

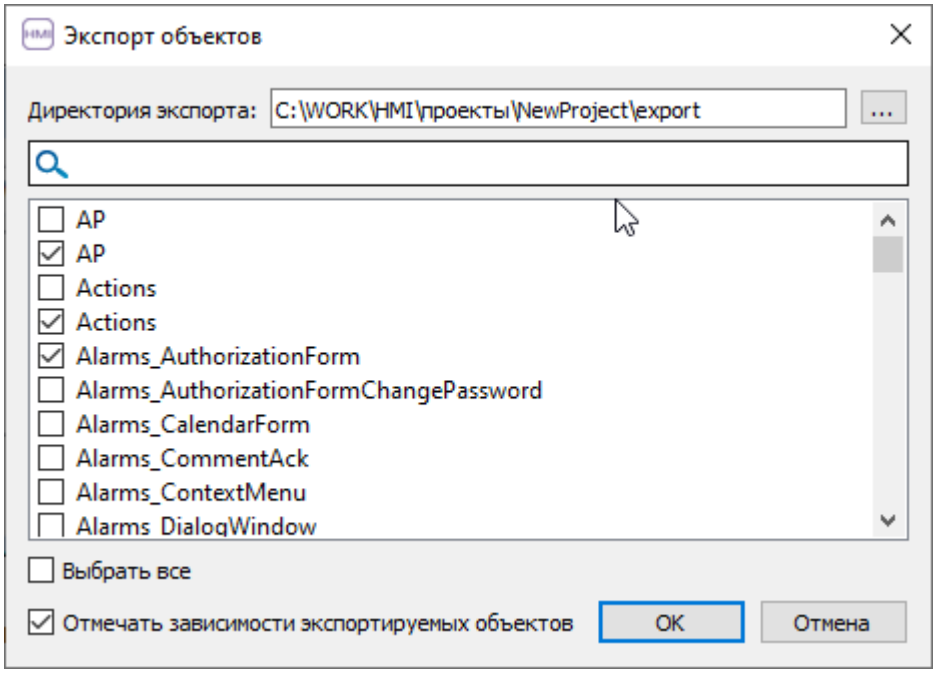


5.2. Импорт и экспорт объектов и экранных форм

В дизайнере SePlatform.HMI вы можете экспортировать и импортировать экранные формы или типы графических объектов в формате `*.omobj`.

Чтобы экспортировать типы графических объектов или формы:

1. Выберите команду **Файл** → **Экспорт объектов**.
2. В окне **Экспорт объектов** выберите директорию экспорта и типы, которые должны быть экспортированы.



По указанной директории экспортируются файлы выбранных типов.

Чтобы импортировать типы графических объектов или формы:

1. Выберите команду **Файл** → **Импорт объектов**.
2. В окне **Импорт объектов ОМ** выберите типы и формы, которые необходимо добавить к уже существующим в проекте.

> ПРИМЕЧАНИЕ
> Чтобы выделить несколько объектов, используйте клавиши **«Ctrl»** и **«Shift»**.

Если импортированные типы и формы уже существуют в проекте, дизайнер SePlatform.HMI предложит сохранить их копии.

5.3. Использование глобальных объектов

Глобальные объекты в SePlatform.HMI представляют собой объекты, к которым можно обращаться из любой части проекта. Это значительно упрощает работу с множеством экранных форм и объектов, которые используют общие ресурсы. Примером может служить централизованное подключение к серверу по протоколу TCP. Используя глобальные объекты, можно избежать необходимости повторной настройки подключения к серверу в каждой экранной форме.

5.3.1. Пример использования глобальных объектов

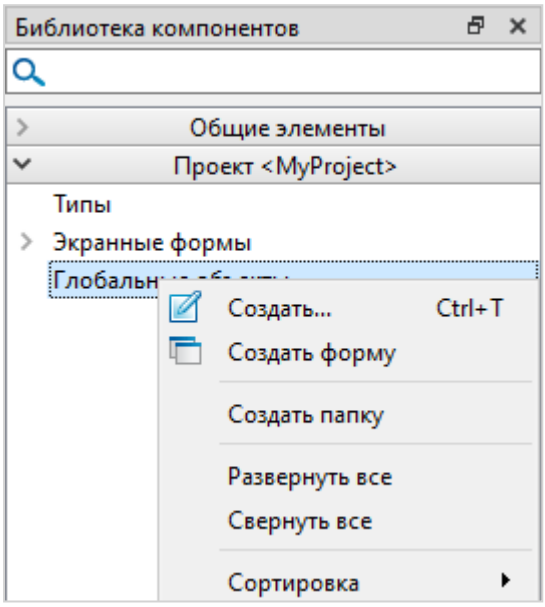
Далее кратко описан пример использования глобального объекта для централизованного подключения к серверу по протоколу TCP в проекте.

В примере показано, как:

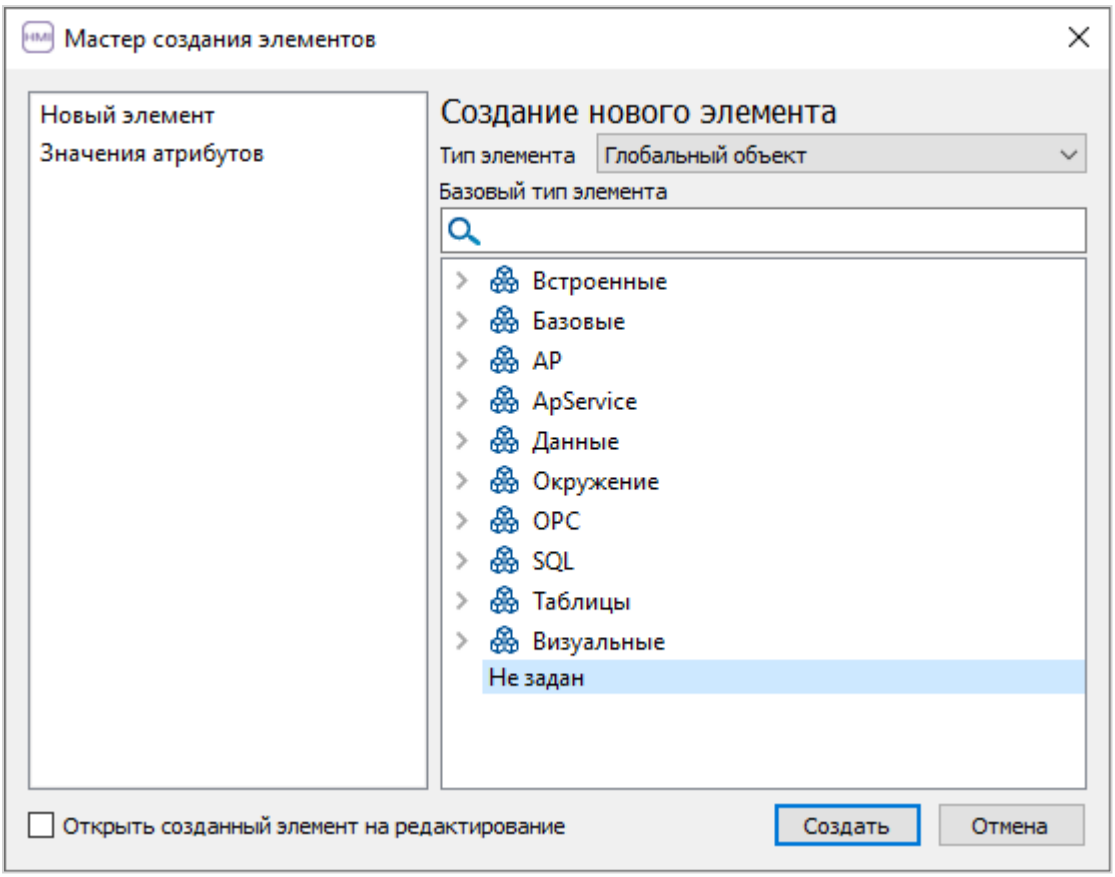
- Создать глобальный объект;
- Создать копию имеющегося глобального объекта;
- Сгруппировать глобальные объекты;
- Обратиться к глобальному объекту из другой части проекта.

5.3.1.1. Создать глобальный объект

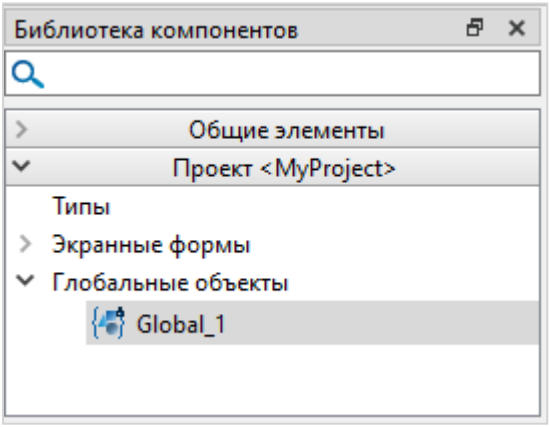
1. Чтобы создать новый глобальный объект, в **Библиотеке компонентов** во вкладке **Проект** щёлкните правым кликом мыши на **Глобальные объекты** → **Создать**.



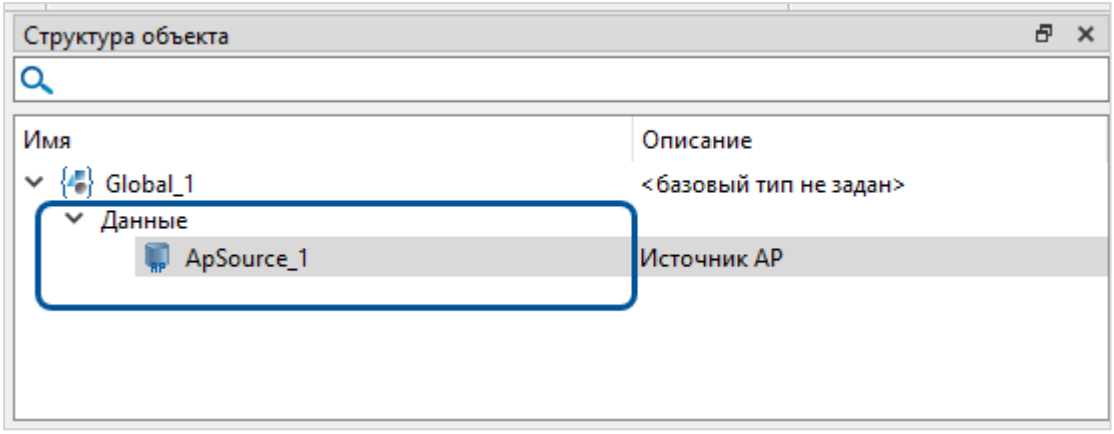
2. Откроется **Мастер создания элементов**. В поле **Тип элемента** выберите **Глобальный объект**, **Базовый тип элемента - Не задан**.



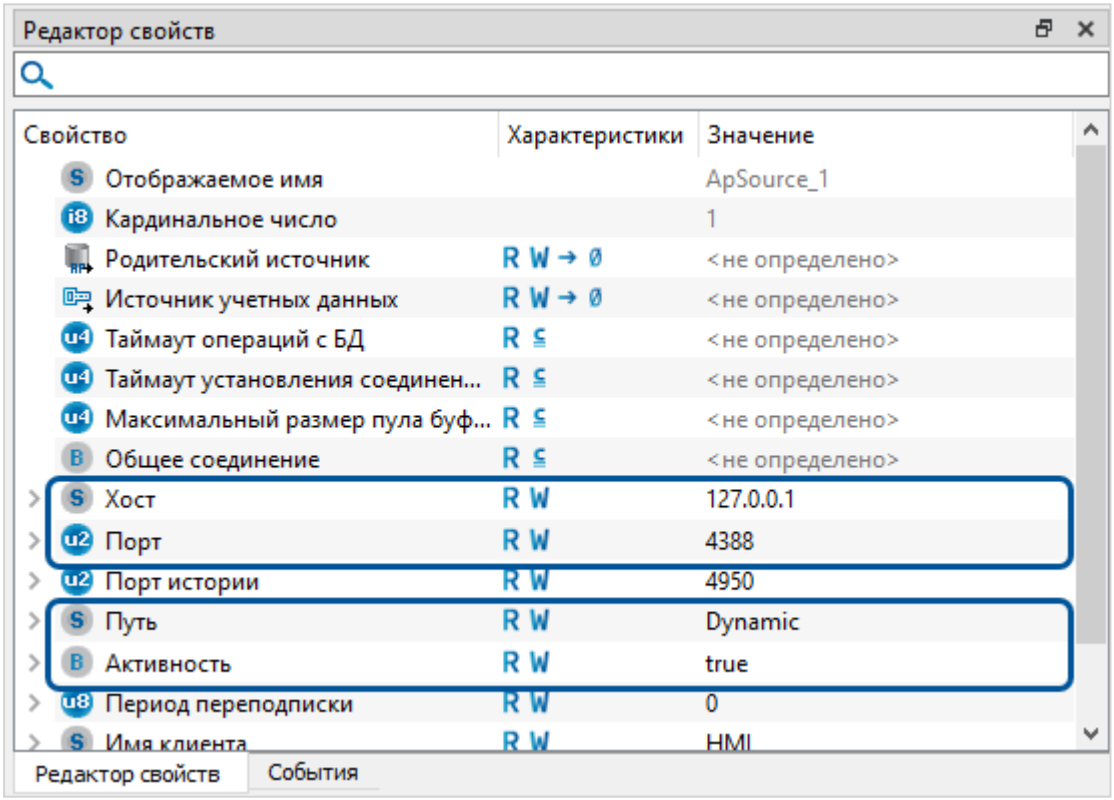
3. Создастся новый глобальный объект со стандартным названием. Дважды щелкните по созданному глобальному объекту для открытия его рабочей области.



4. Добавьте на экранную форму компонент **Источник AP**, чтобы взаимодействовать с источником данных по протоколу TCP.



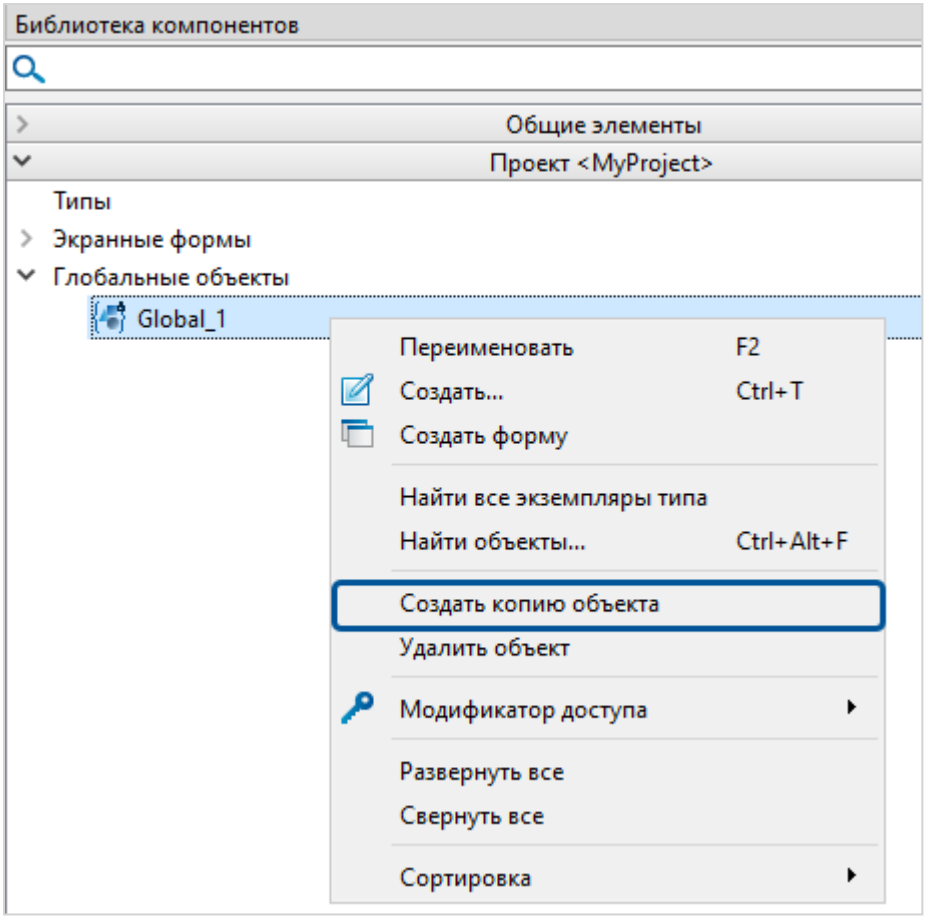
5. Чтобы компонент мог взаимодействовать с источником данных, настройте свойства Хост, Порт, Путь и Активность



Теперь на этот источник данных можно ссылаться из любого места в проекте.

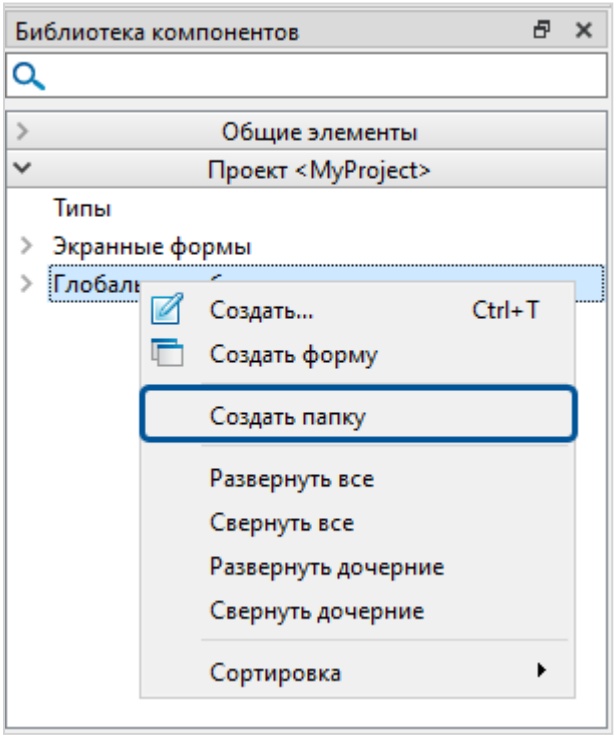
5.3.1.2. Создать копию имеющегося глобального объекта

Чтобы создать копию имеющегося глобального объекта, в **Библиотеке компонентов** выберите глобальный объект и в контекстном меню объекта выберите **Создать копию объекта**.



5.3.1.3. Сгруппировать глобальные объекты

Чтобы группировать глобальные объекты, используйте папки. Для создания папки перейдите в контекстное меню и выберите команду **Создать папку**.



ОБРАТИТЕ ВНИМАНИЕ

Избегайте совпадения имён объектов на разных уровнях иерархии. Например, если имя объекта в папке совпадёт с именем объекта вне папки, то вызываться будет объект размещённый вне папки.

5.3.1.4. Обратиться к глобальному объекту из другой части проекта

К глобальному объекту можно обращаться из любого объекта или экранной формы без непосредственного размещения глобального объекта на ней. Для обращения к глобальному объекту из кода, вы можете использовать ключевое слово **«unit»** или обойтись без него.

Пример обращения к Источнику данных AP глобального объекта Global_1:

- с использованием ключевого слова **«unit»**:

```
unit.Global_1.ApSource_1;
```

- без использования ключевого слова **«unit»**:

```
Global_1.ApSource_1;
```

ОБРАТИТЕ ВНИМАНИЕ

Чтобы обратиться к глобальному объекту, расположенному в папке, в коде указывайте только имя объекта без имени папки.

6. Работа со свойствами объектов

Свойства - это атрибуты, которые определяют особенности поведения или отображения объекта на экране (например, **ширина**, **высота** или **цвет заливки** для графического объекта). Свойства выделенного объекта отображаются в области **Редактор свойств**. Пользователь может создавать свои собственные свойства для любых объектов.

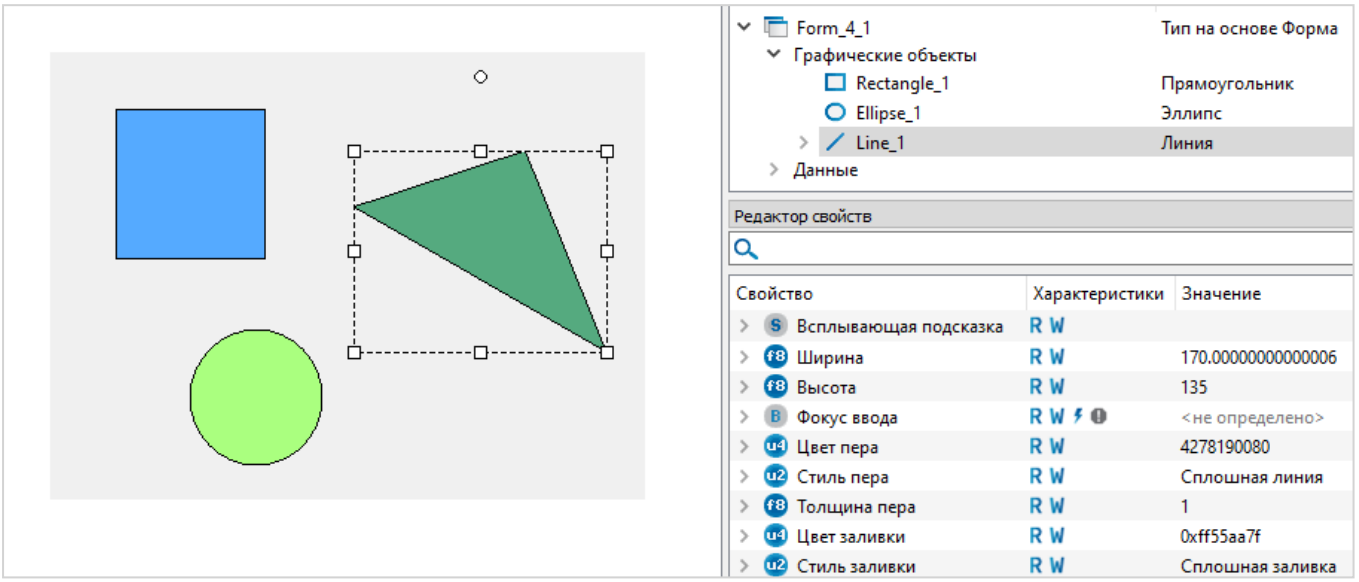
Любое свойство является контейнером для данных определенного типа:

Тип	Иконка	Описание и диапазон значений
bool		Логическое значение. Принимает значения **«True»** или **«False»**
int1		Знаковое целое размером 1 байт. Принимает значения от **«-128»** до **«127»**
uint1		Беззнаковое целое размером 1 байт. Принимает значения от от **«0»** до **«255»**
int2		Знаковое целое размером 2 байта. Принимает значения от **«-32 768»** до **«32 767»**
uint2		Беззнаковое целое размером 2 байта. Принимает значения от **«0»** до **«65535»**
int4		Знаковое целое размером 4 байта. Принимает значения от **«-2 147 483 648»** до **«2 147 483 647»**
uint4		Беззнаковое целое размером 4 байта. Принимает значения от **«0»** до **«4 294 967 295»**
int8		Знаковое целое размером 8 байт. Принимает значения от **«$-9*10^{18}$»** до **«$9*10^{18}$»**
uint8		Беззнаковое целое размером 8 байт. Принимает значения от **«0»** до **«18 446 744 073 709 551 615»**
float		Значение с плавающей запятой размером 4 байта
double		Значение с плавающей запятой размером 8 байт
string		Строковое значение
timestamp		Дата, время
variant		Универсальный тип данных. Свойство этого типа может хранить любые значения. Используется, когда заранее не известен род данных, с которыми предстоит работать.

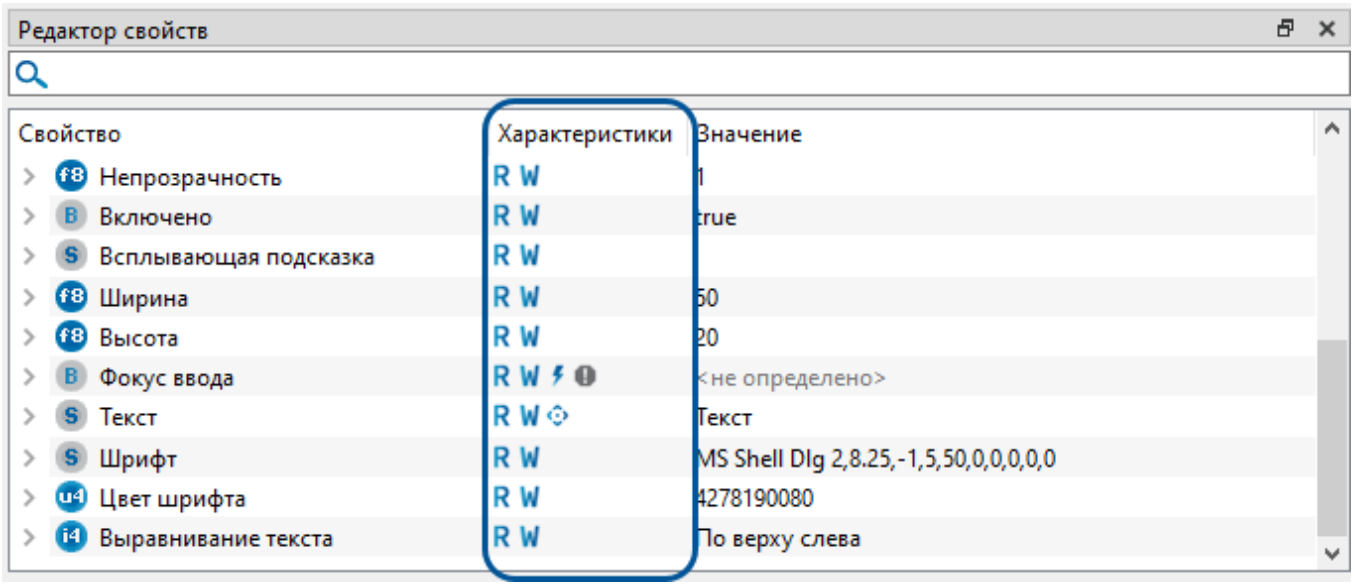
В зависимости от момента определения, свойства делятся на свойства режима проектирования (стр. 67) (design time) и свойства режима исполнения (стр. 69) (run-time).

6.1. Свойства в режиме проектирования

Набор свойств, определяющих состояние объекта в Дизайнере SePlatform.HMI. Чтобы установить значения свойств объекта, выделите объект и перейдите в **Редактор свойств**. На рисунке ниже показан треугольник и его свойства в режиме проектирования.



В **Редакторе свойств** каждое свойство обладает набором характеристик, определяющих его поведение и возможности взаимодействия с ним.

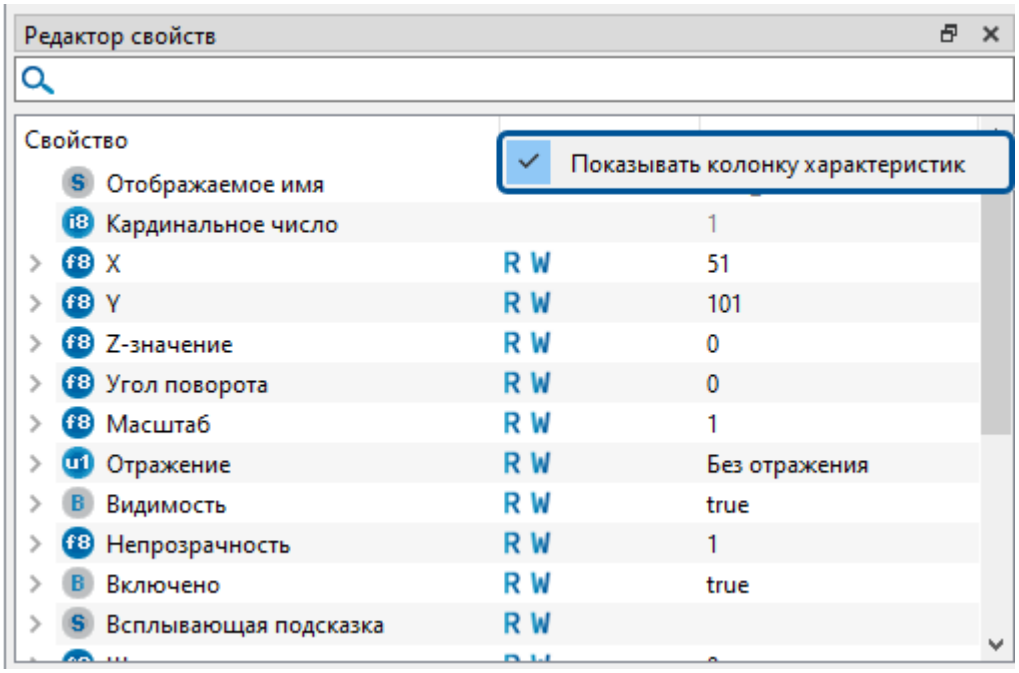


Ниже представлена таблица характеристик элементов свойств.

Тип характеристики	Иконка	Описание
«Константный (неизменяемый)»		Свойство устанавливается один раз и не может быть изменено в режиме рантайма.

Тип характеристики	Иконка	Описание
«Ссылка»		Свойство содержит ссылку на другой объект в проекте.
«Обнуляемый»		Свойство может быть сброшено, позволяя убрать установленную ссылку.
«Доступен для чтения»		Свойство может быть прочитано, но не изменено в режиме рантайма.
«Доступен для записи»		Свойство может быть изменено в режиме рантайма.
«Уведомляющий»		Свойство, изменение которого генерирует уведомления или вызывает определённые события.
«Параметр инициализации»		Свойство используется для инициализации типов и не может быть изменено в режиме рантайма. Позволяет отличить параметры инициализации от обычных ссылок.
«Значение задано»		Для свойства установлена ссылка или задано значение.
«По умолчанию»		Свойство, к которому можно обращаться напрямую через объект, без необходимости указывать имя свойства. Например, для объекта **«Text_1»** со свойством Text, запись Text_1 = "qwer" эквивалентна записи Text_1.Text = "qwer".
«Элемент устарел»		Свойство устарело и может быть удалено в будущих версиях.

Чтобы скрыть или показать столбец характеристик в **Редакторе свойств**, кликните правой кнопкой мыши по названию столбца **«Характеристики»** и выберите **«Показывать колонку характеристик»** в выпадающем меню.



6.2. Свойства в режиме исполнения

Набор свойств, определяющих состояние объекта после запуска проекта в рантайм.

Некоторые свойства объектов нередактируемые (только для чтения). К таким свойствам можно обращаться в обработчиках событий или функциях. Полный список свойств приведен в **Справочнике**. Если попытаться скомпилировать проект, где есть попытка установить значения нередактируемым свойствам, то компилятор выдаст ошибку.

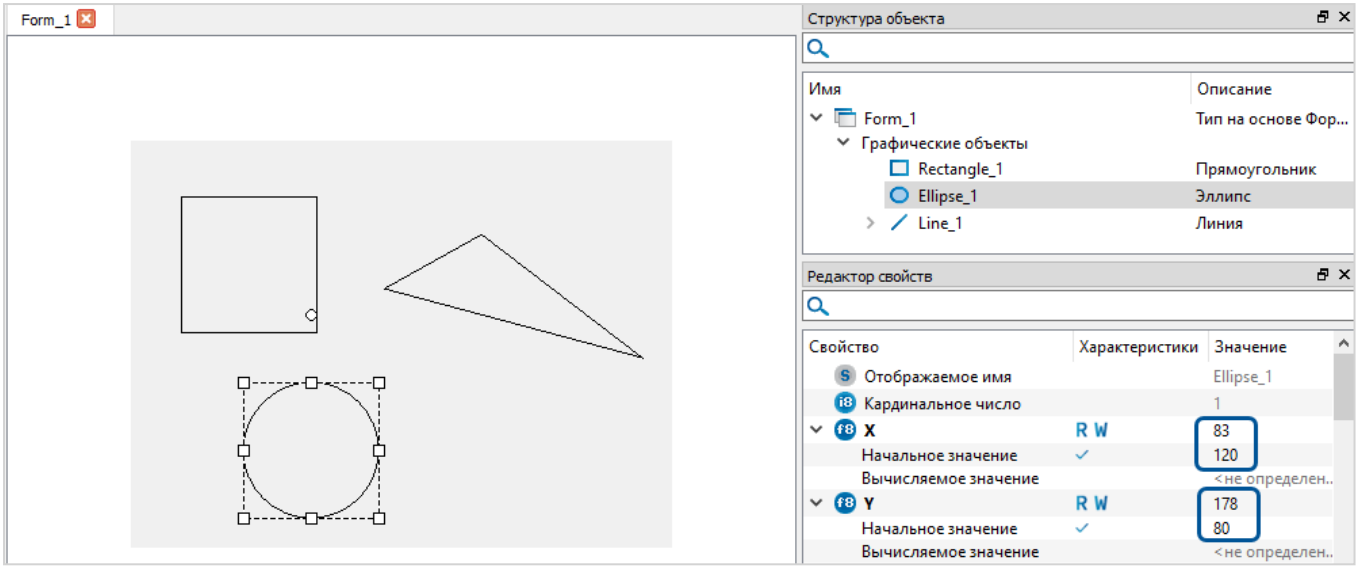
ПРИМЕР

Попытка присвоить значение нередактируемому свойству

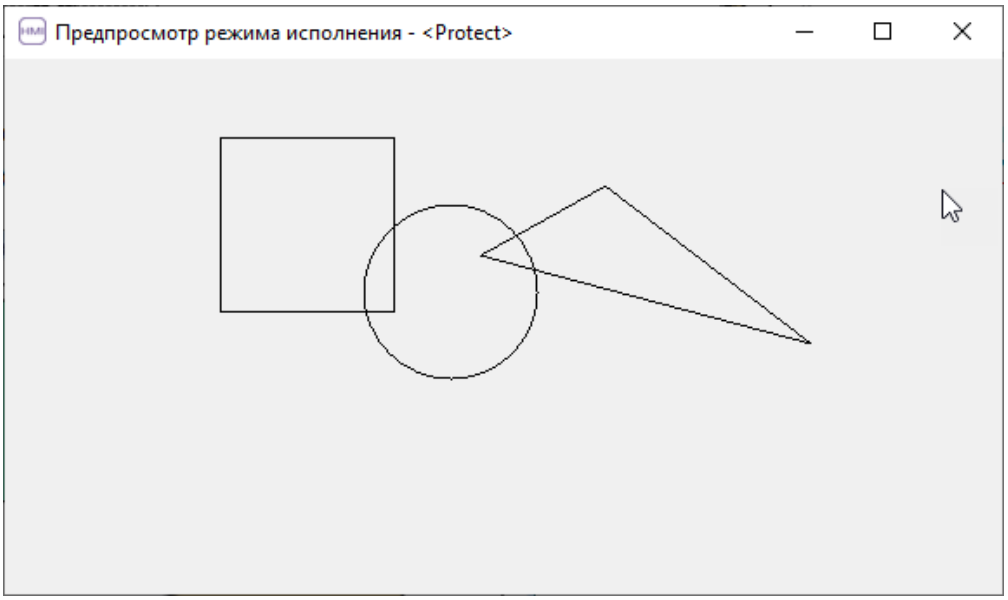
```
OpcItem.Quality = 192;
```

6.2.1. Начальные значения свойств

Начальные значения свойств объектов применяются в момент инициализации экранной формы в рантайме. Чтобы указать начальное значение свойства для объекта, введите параметр в поле Начальное значение. На рисунке ниже показано, что начальные значения свойств (X и Y для круга) отличаются от значений этих же свойств в режиме проектирования.



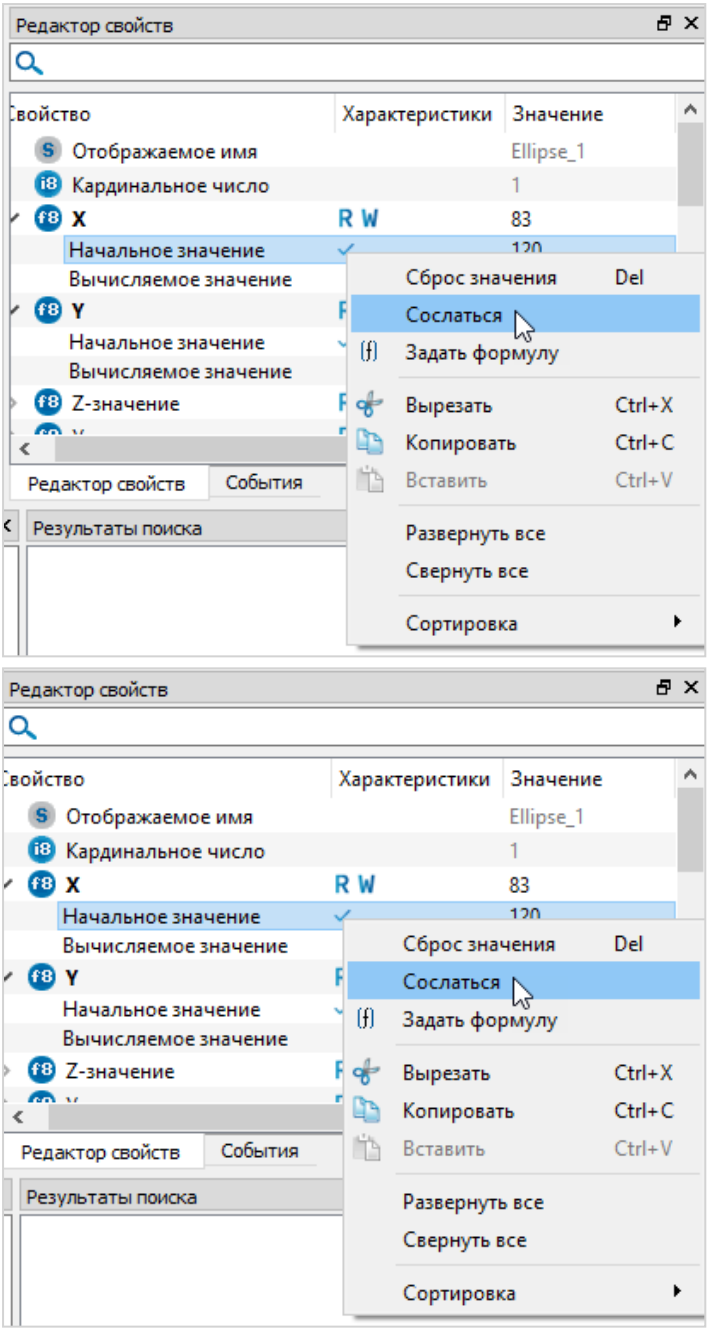
На рисунке ниже показано, как после запуска в рантайме, объекты экранной формы были инициализированы начальными значениями свойств, и круг занял позицию в соответствии с начальными значениями свойств X и Y.



Если **Начальное значение** для свойства не указано, то для целей инициализации в рантайме берется значение свойства в режиме проектирования (стр. 67).

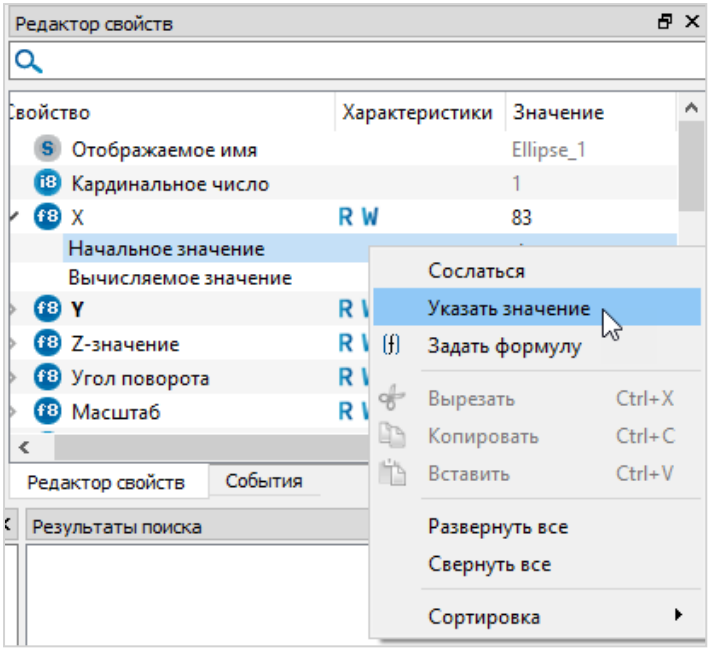
6.2.1.1. Сослаться на значение

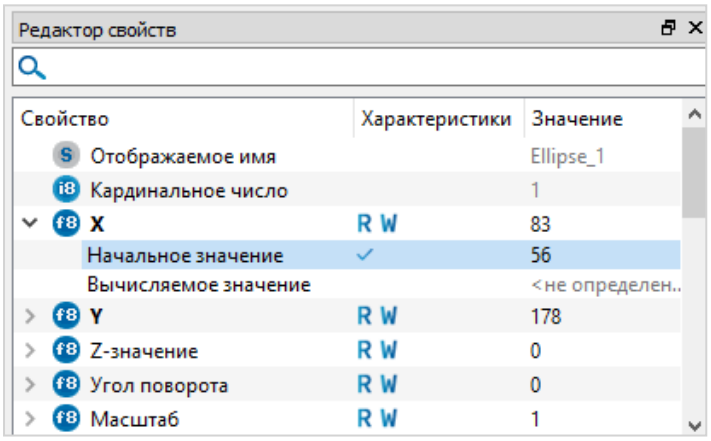
Начальное значение свойства может ссылаться на значение ранее инициализированного параметра (стр. 58). Для этого в контекстном меню выберите команду **Сослаться** и укажите нужный параметр инициализации.



6.2.1.2. Указать значение

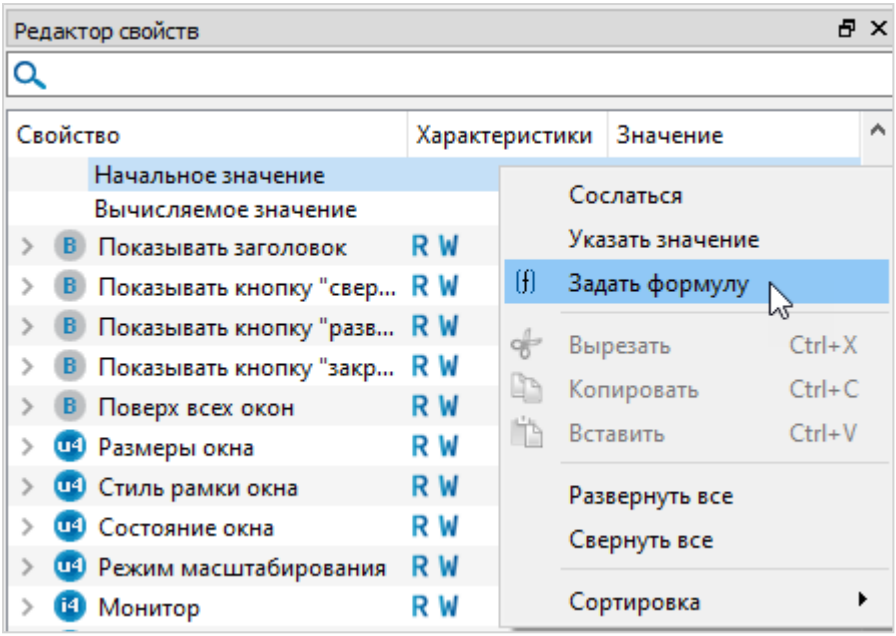
Для указания начального значения свойства в контекстном меню выберите команду **Указать значение** и введите нужное значение.





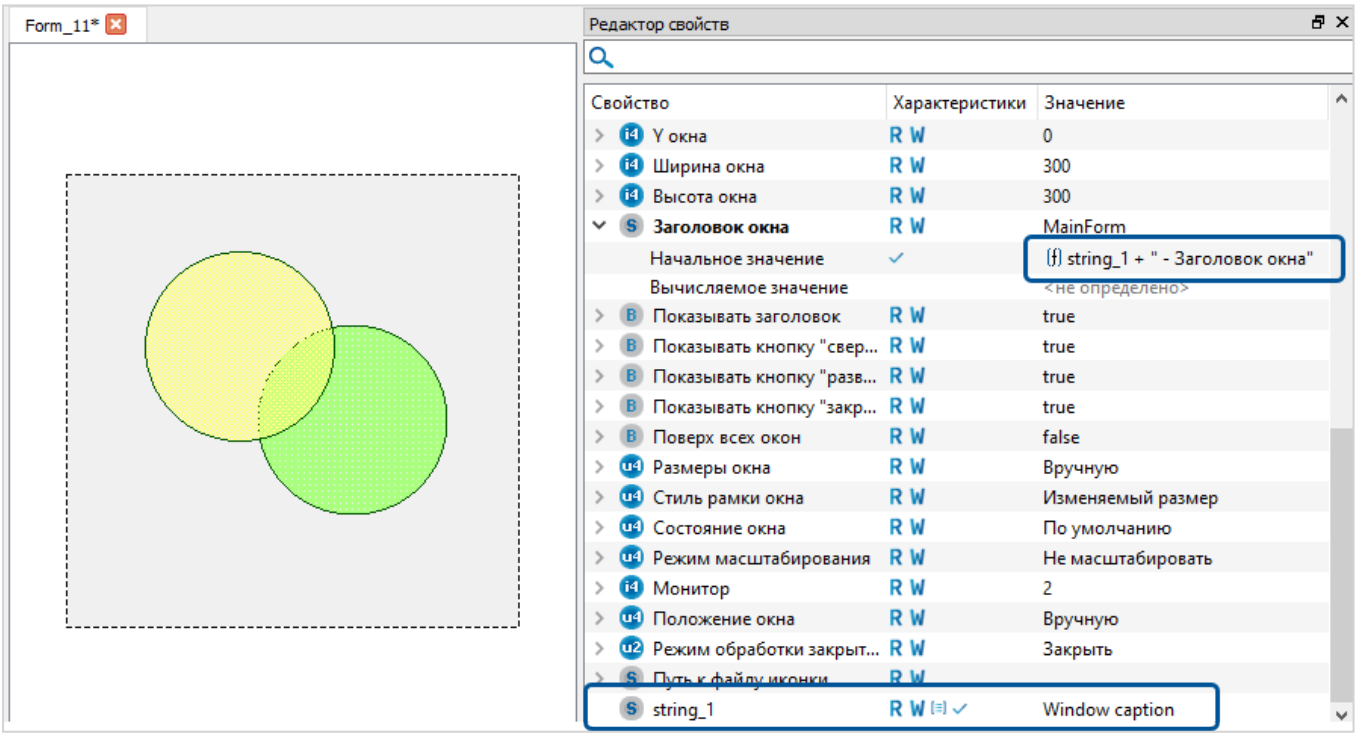
6.2.1.3. Задать формулу

Начальное значение свойства может быть указано в виде формулы. Формула может состоять из одного или нескольких параметров инициализации объектов (стр. 58) и/или строк. Чтобы начальное значение задать в виде формулы, в контекстном меню выберите команду **Задать формулу** и укажите нужную формулу.

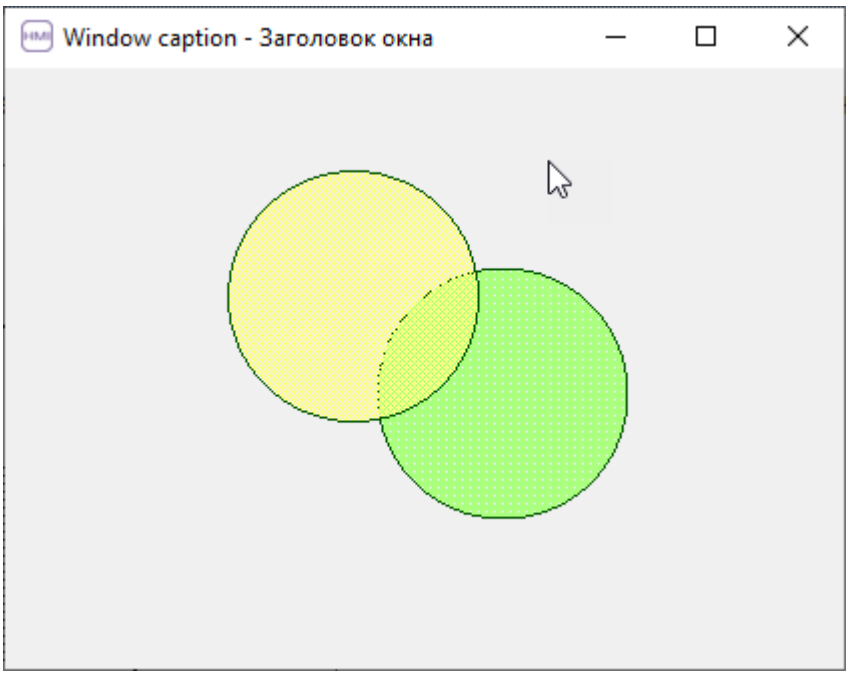


В примере ниже показано, как в заголовок окна формы выводится формула, состоящая из параметра инициализации типа string и строкового значения **«Заголовок окна»**:

```
string_1 + " - Заголовок окна"
```



При запуске формы в рантайме заголовок формы будет содержать заданную в начальном свойстве формулу.



6.2.2. Вычисляемые значения свойств

Значения свойств могут вычисляться в процессе работы приложения по формуле, при наступлении определенных условий.

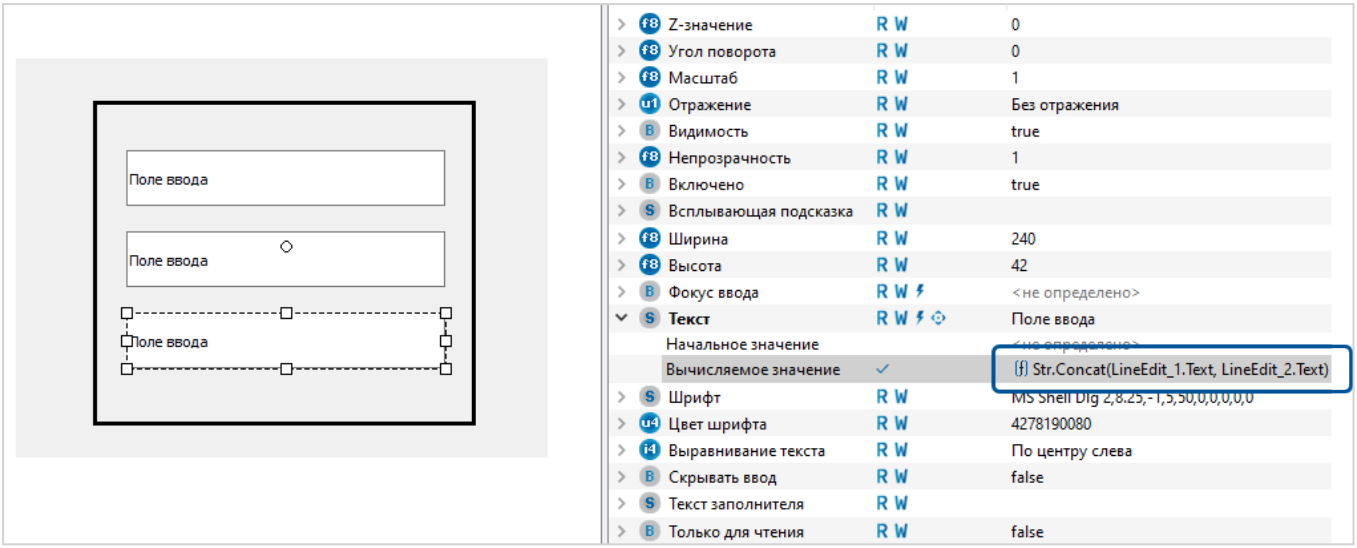
6.2.2.1. Формула

Значения свойств могут динамически вычисляться в рантайме по логике, заданной в формуле - выражении на языке SePlatform.Om.

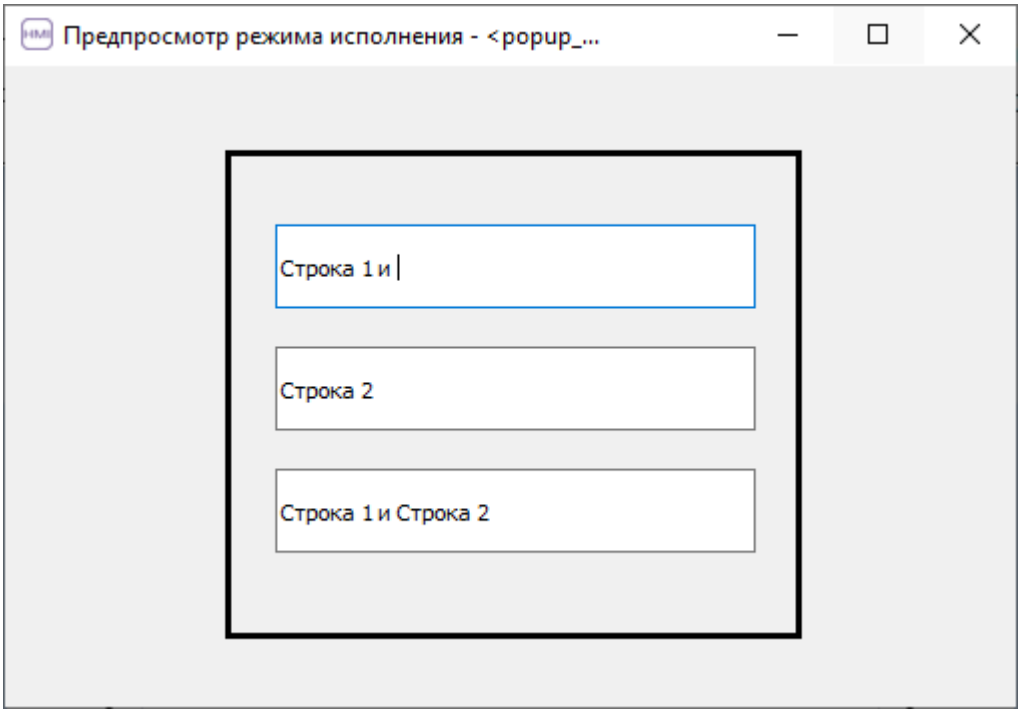
В формулу могут передаваться значения свойств собственных объектов формы, а также объектов других форм. В текущей версии SePlatform.HMI возможна передача значений только из главной формы в дочерние.

В примере ниже показано, как свойство **Текст** (для нижнего поля ввода) вычисляется по формуле (применена стандартная функция языка SePlatform.Om для конкатенации двух строк):

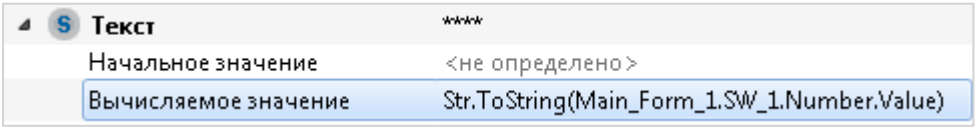
```
Str.Concat(TextEdit_1.Text, TextEdit_2.Text)
```



По логике, заданной в формуле, текст нижнего поля будет отображать результат слияния текста из верхних полей. Результат будет пересчитываться динамически при изменении любого аргумента формулы.



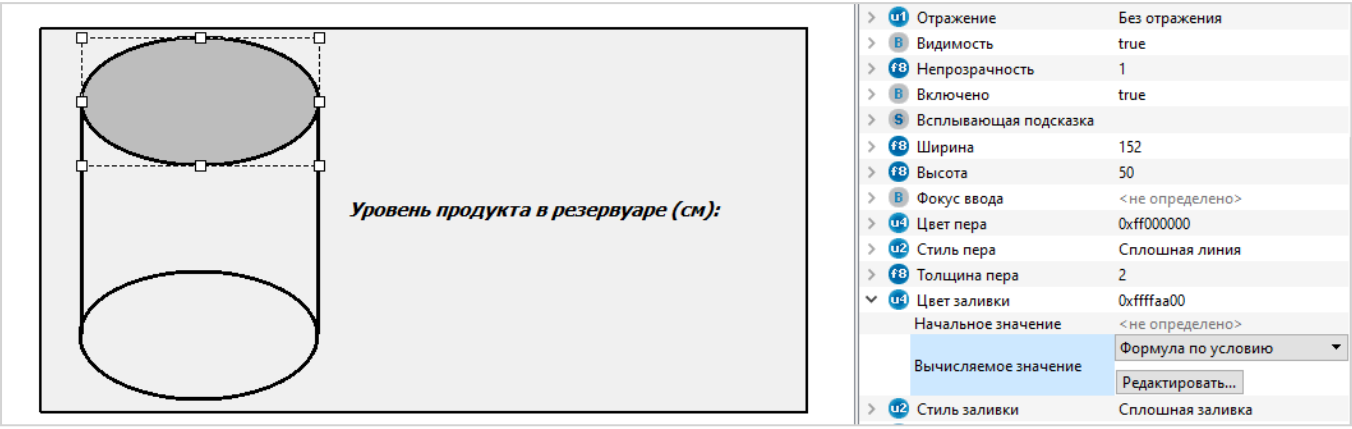
Для подписки на свойства объектов других форм, обращайтесь к объектам через ссылку на родительскую форму. На рисунке ниже показано обращение к значению элемента **«Number»**, который принадлежит родительской форме, через ссылку на родительскую форму **«Main_Form_1»**.



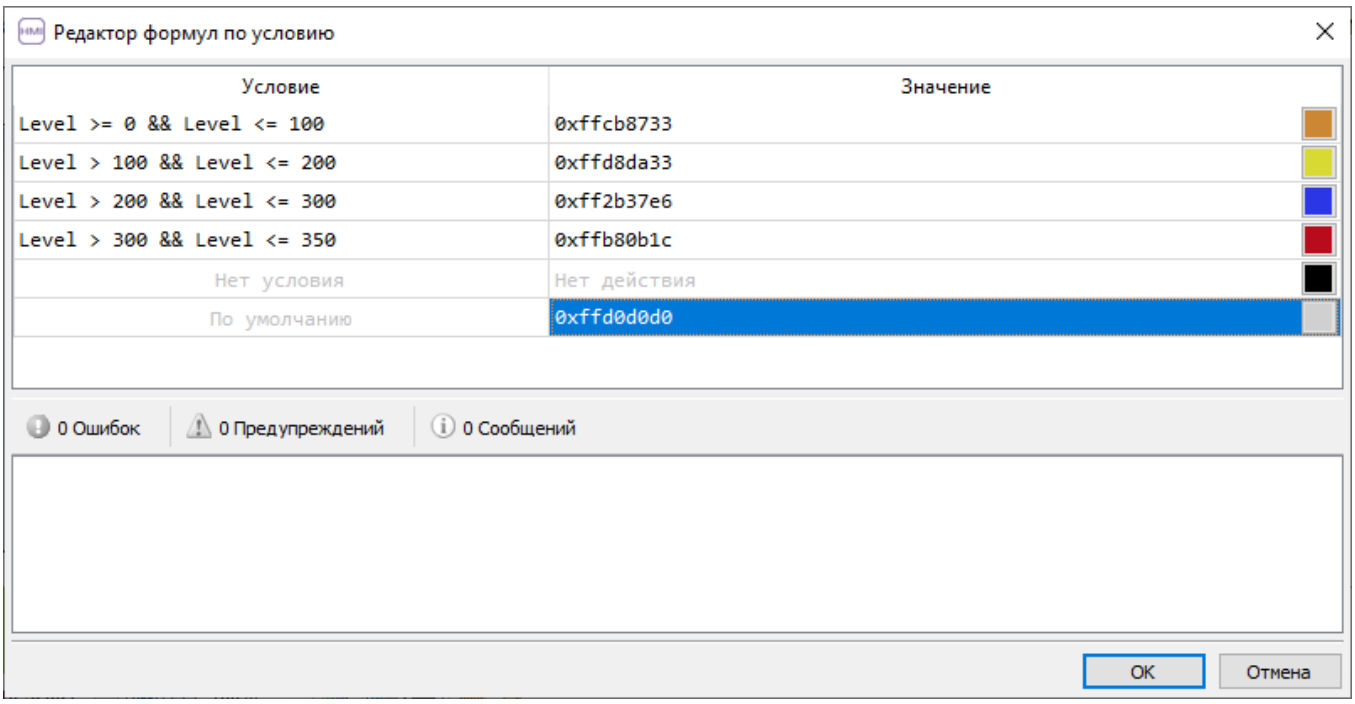
6.2.2.2. Формула по условию

Формулы по условию применяются для выставления значений свойствам в рантайме в зависимости от выполнения некоторых условий. На рисунках ниже показан резервуар, верхушка которого окрашивается в

определенный цвет в зависимости от уровня нефти. Для установки формулы по условию, выберите для свойства **Вычисляемое значение** типа «**Формула по условию**» и нажмите кнопку **Редактировать**.



В появившемся окне **Редактор формул по условию** установите нужные условия и результирующее значение при выполнении каждого условия. Поле **По умолчанию** содержит значение, которое будет выставляться, если ни одно условие не было удовлетворено.



ПРИМЕЧАНИЕ

Чтобы быстро заполнять таблицу редактора формул по условию, используйте горячие клавиши:

- «**Ctrl**»+«**C**» (скопировать строку);
- «**Ctrl**»+«**V**» (вставить строку);
- «**Ctrl**»+«**Shift**»+(«+») (вставить строку со сдвигом).

После запуска в рантайме, заливка эллипса будет динамически меняться в зависимости от уровня продукта в резервуаре (значение «**Level**»).


Уровень продукта в резервуаре (см): 50

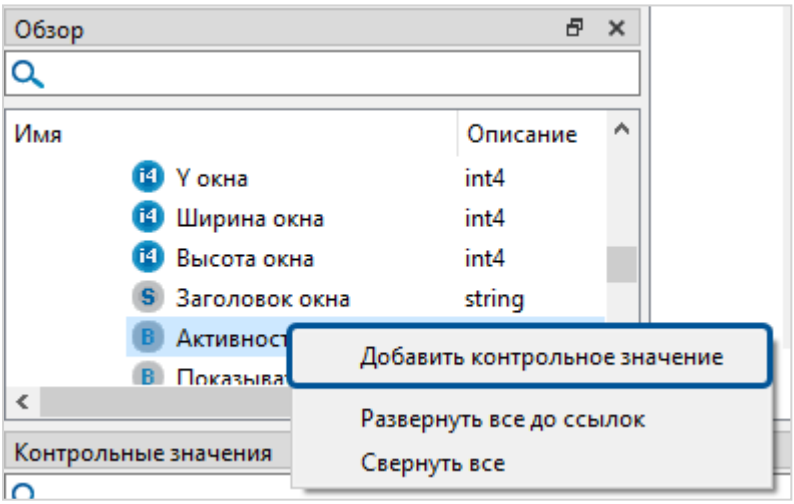

Уровень продукта в резервуаре (см): 150


Уровень продукта в резервуаре (см): 250

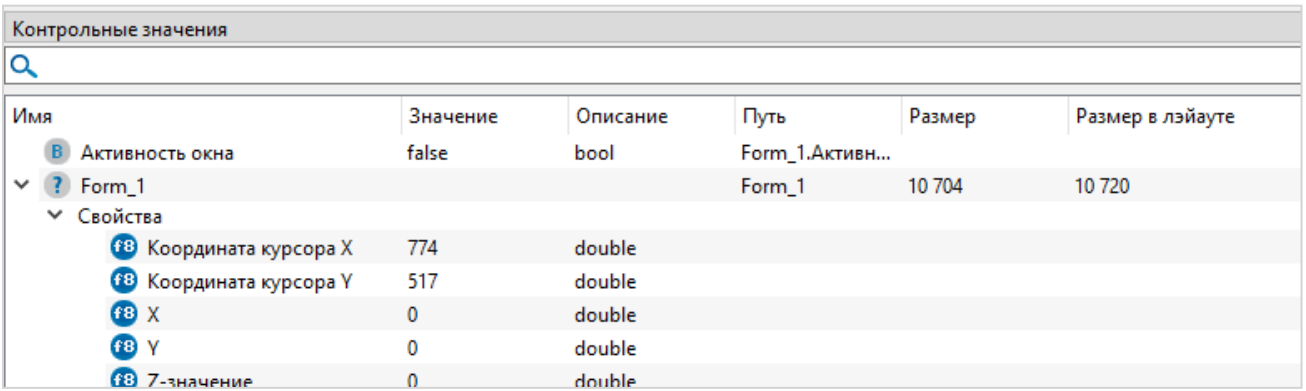
6.2.3. Отслеживание и редактирование значений свойств

Чтобы просмотреть и изменить текущие значения свойств объектов:

1. Запустите проект или форму в режим рантайма.
2. Перейдите в окно **Обзор** (появляется только при наличии запущенной на исполнение формы), выберите контролируемый объект или свойство, вызовите контекстное меню и выполните команду **Добавить контрольное значение**.



3. В окне **Контрольные значения** (появляется только при наличии запущенной на исполнение формы) отображаются текущие значения свойств выбранного объекта.



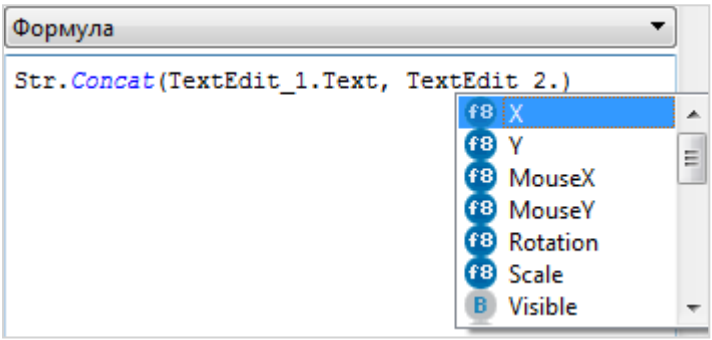
Чтобы изменить значения свойств редактируйте значения в окне **Контрольные значения**. Свойства, доступные только для чтения изменять нельзя.

6.3. Работа со свойствами из скриптов

Чтобы обращаться к свойствам объектов из скриптов и формул, используйте точку в качестве разделителя между именем объекта и именем свойства:

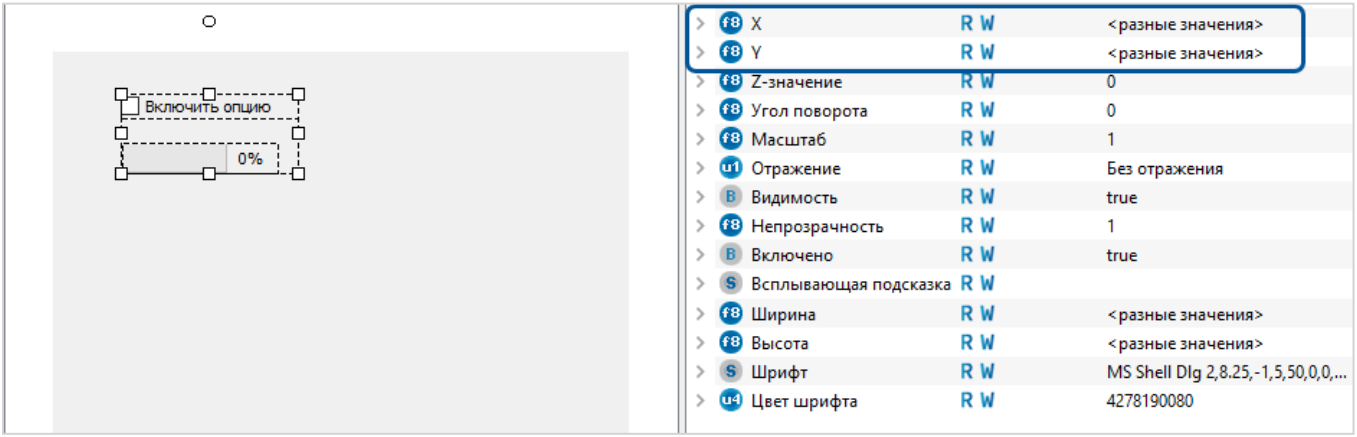
```
//Установить значение 1 свойству Property для объекта Object
Object.Property = 1;
//Присвоить переменной Variable значение свойства Property объекта Object
Variable = Object.Property;
```

В процессе написания скриптов или формул возможные варианты свойств объекта можно выбирать из выпадающего списка, который появляется после ввода точки.



6.4. Групповое изменение свойств объектов

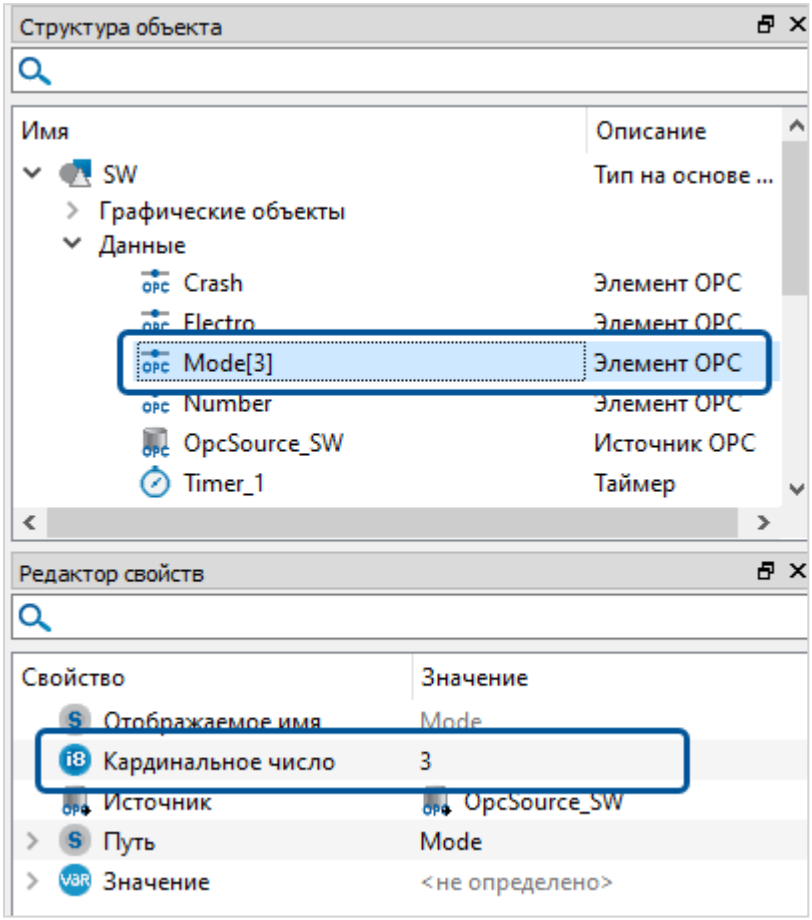
Чтобы массово поменять схожие свойства у множества объектов, выделите их на рабочей области или в области **Структура объектов**. После выделения нескольких объектов в области **Редактор свойств** останутся лишь схожие свойства. На рисунке ниже показано как выделены объекты **Индикатор прогресса** и **Флажок**, а в редакторе свойств остались только схожие свойства этих объектов. Различающиеся значения свойств отмечены фразой разные значения.



6.5. Массивы

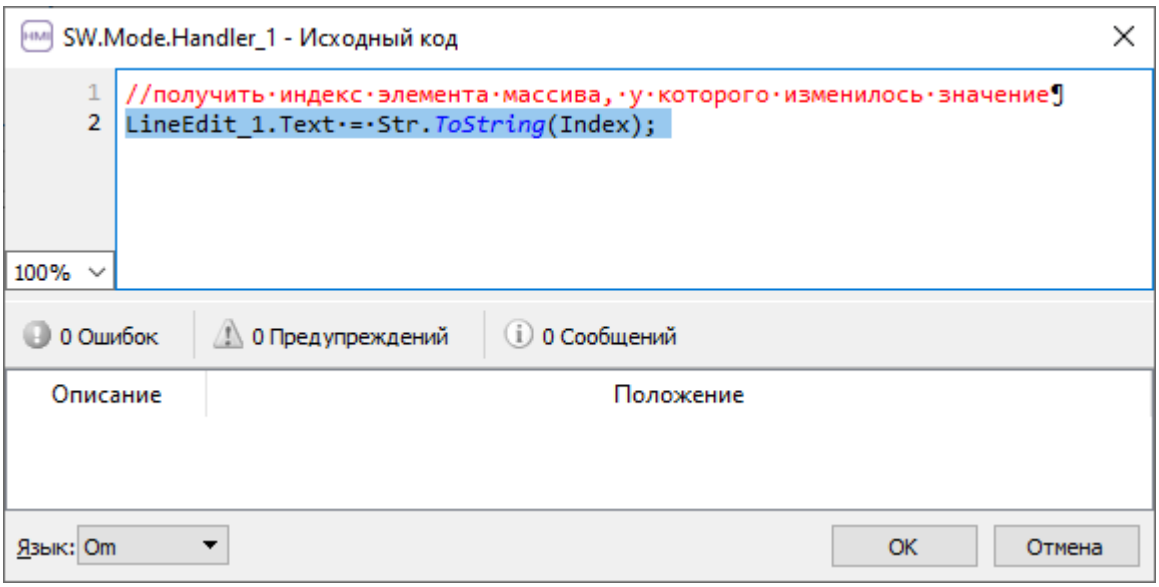
Любой объект в SePlatform.HMI (элемент из библиотеки или объект, созданный из разных элементов) можно преобразовать в массив объектов. Массивы позволяют хранить несколько объектов внутри одного объекта, добавленного на форму. Все объекты в массиве (элементы массива) имеют один тип значений.

Чтобы преобразовать объект в массив, укажите размер массива (количество элементов в массиве) в свойстве **Кардинальное число** объекта. Достаточно указать значение больше «1».

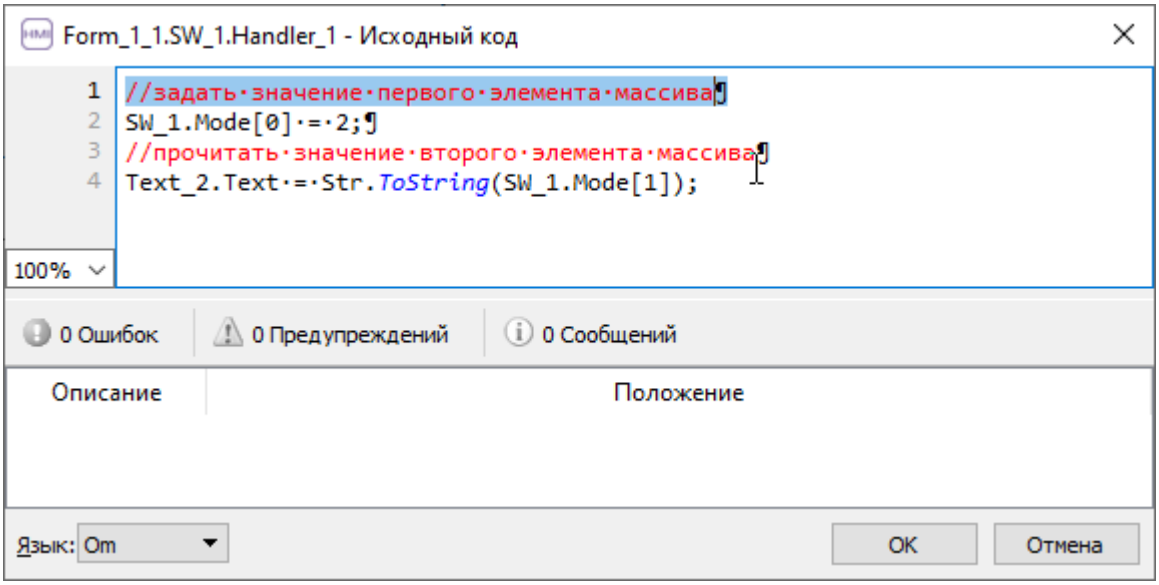


Каждый элемент в массиве имеет свой индекс. Нумерация элементов массива начинается с 0.

Чтобы получить индекс элемента, у которого сработало какое-то событие, используйте свойство **Index** объекта-массива.



Чтобы обращаться к элементам массива, указывайте индекс элемента в квадратных скобках после имени объекта-массива.



6.5.1. Пример взаимодействия с массивами

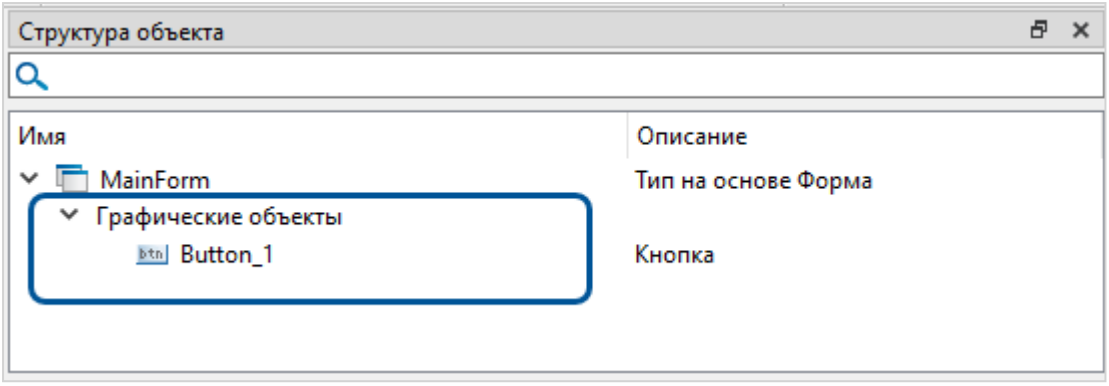
Далее кратко описан пример (файлы проекта идут в комплекте с документацией) взаимодействия с массивом кнопок в проекте SePlatform.HMI с использованием языка SePlatform.Om.

В примере показано, как:

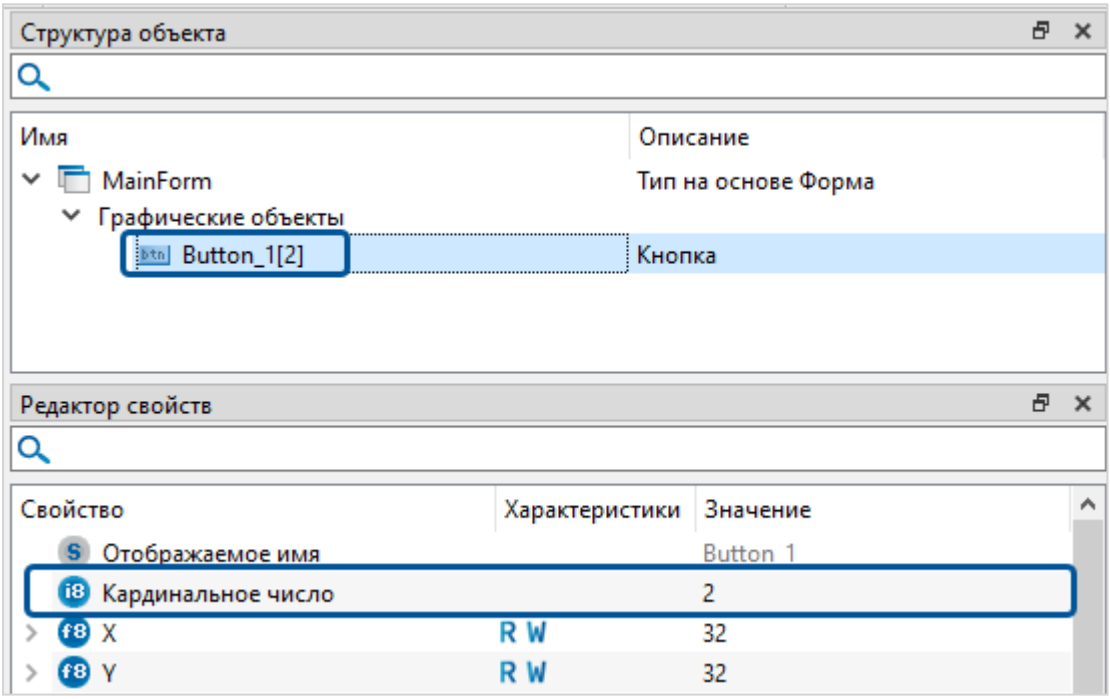
- Создать массив из двух кнопок;
- Изменить координаты второй кнопки;
- Установить зеленый цвет для нажатой кнопки;
- Отобразить в текстовом поле, какая из двух кнопок была нажата.

6.5.1.1. Создать массив из двух кнопок

1. Добавьте на экранную форму объект типа **Кнопка**.



2. Для организации нескольких кнопок в единый массив настройте свойство Кардинальное число объекта. Для массива из двух кнопок установите значение «2». В структуре объекта автоматически добавляется к имени элемента размер массива в квадратных скобках.



6.5.1.2. Изменить координаты второй кнопки

Чтобы переместить вторую кнопку в новые координаты перед открытием формы в рантайме, настройте обработчик события Opened:

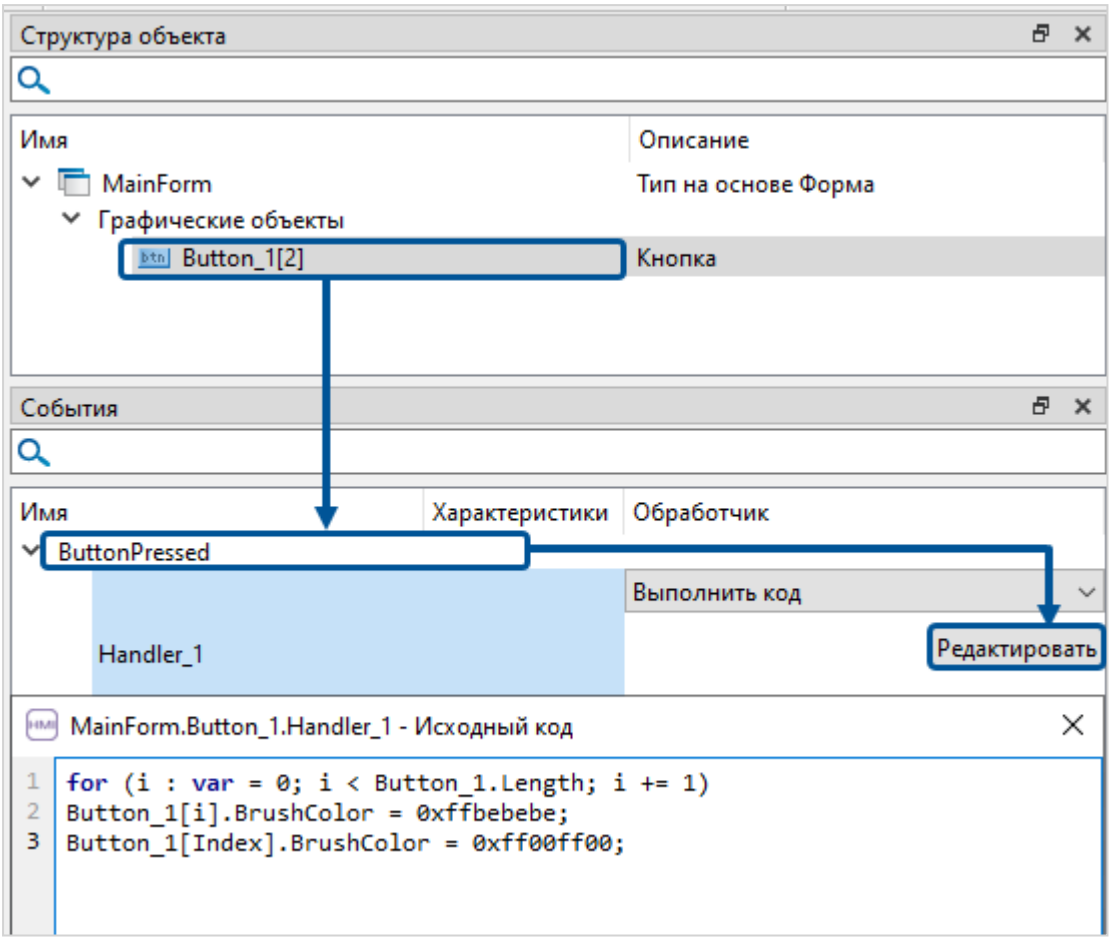
```
Button_1[1].X = 170;
Button_1[1].Y = 110;
```

В данном коде «`Button_1[1]`» обращается к определенному элементу в массиве кнопок по его порядковому номеру, где «`[1]`» соответствует второй кнопке в массиве. Установка новых значений для свойств X и Y перемещает эту конкретную кнопку в указанные координаты на экране.

6.5.1.3. Установить зеленый цвет для нажатой кнопки

Чтобы при нажатии на кнопку она подсвечивалась зеленым цветом, настройте событие ButtonPressed. В структуре объекта выделите компонент **Кнопка**, перейдите во вкладку **События** и нажмите на событие

ButtonPressed правым кликом мыши→ **Добавить обработчик** → **Выполнить код** → **Редактировать**. В открывшемся окне введите следующий код:



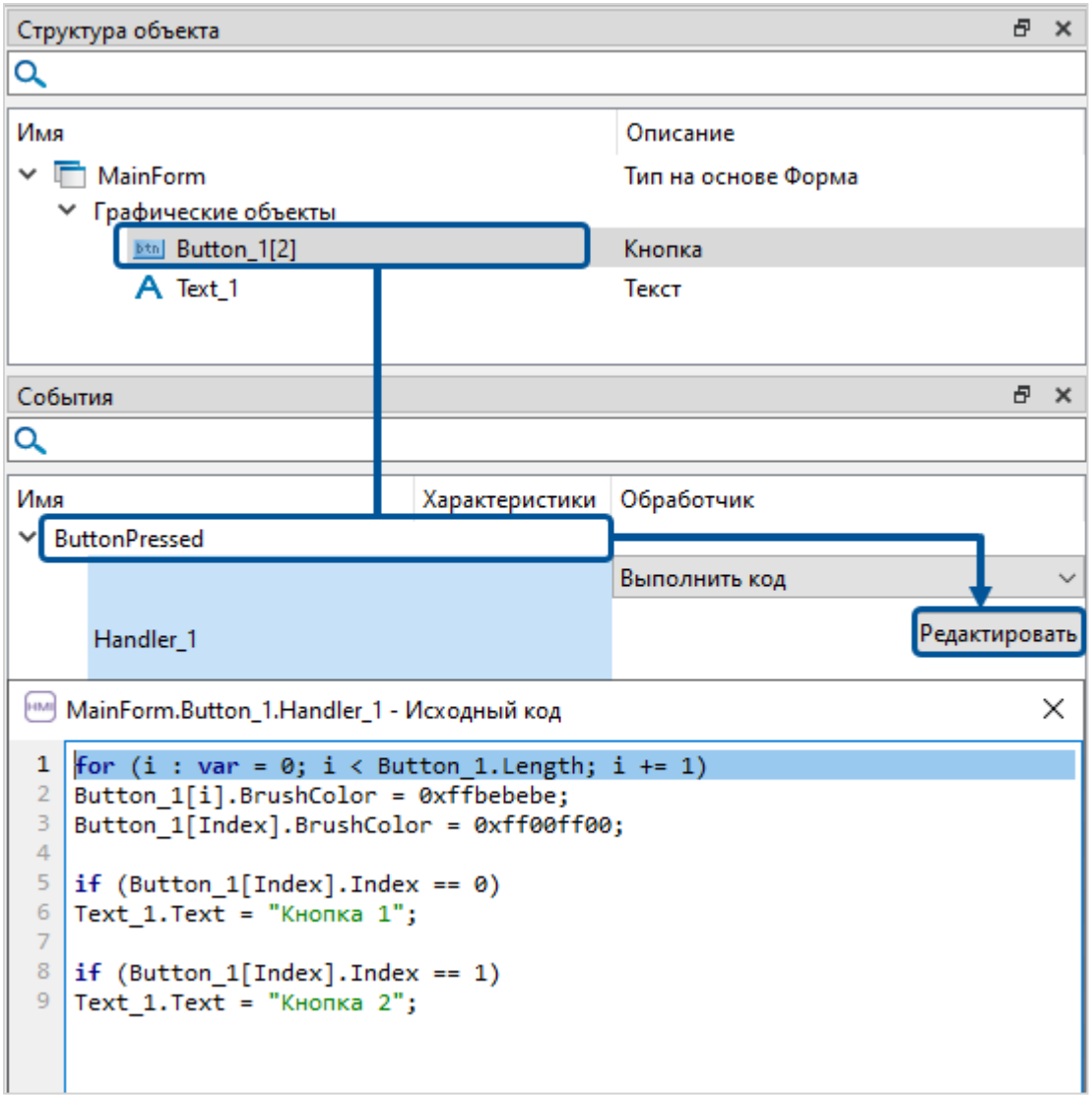

```
for (i : var = 0; i < Button_1.Length; i += 1) // Цикл проходит по всем кнопкам в массиве
Button_1
 Button_1[i].BrushColor = 0xffbebebe; // Устанавливает серый цвет для каждой кнопки

Button_1[Index].BrushColor = 0xff00ff00; // Устанавливает зеленый цвет для нажатой кнопки
```

Этот код осуществляет итерацию по всем кнопкам в массиве **«Button_1»** с помощью цикла (переменная **«i»** - это индекс текущей кнопки). Для каждой кнопки в массиве устанавливается изначальный цвет - серый. После этого устанавливается зеленый цвет для кнопки, которая была нажата (определенной по значению свойства Index).

6.5.1.4. Отобразить в текстовом поле, какая из двух кнопок была нажата

Чтобы отображать информацию о том, какая из двух кнопок была нажата, подготовьте текстовое поле и настройте событие ButtonPressed. В структуре объекта выделите компонент **Кнопка**, перейдите во вкладку **События** и нажмите на событие ButtonPressed правым кликом мыши→ **Добавить обработчик** → **Выполнить код** → **Редактировать**. В открывшемся окне введите следующий код:




```
if (Button_1[Index].Index == 0)
 Text_1.Text = "Кнопка 1";
if (Button_1[Index].Index == 1)
 Text_1.Text = "Кнопка 2";
```

Этот код проверяет значение свойства Index нажатой кнопки. Если Index равен **«0»**, то в текстовое поле **«Text_1»** выводится сообщение **«Кнопка 1»**. Если Index равен **«1»**, то выводится сообщение **«Кнопка 2»**.

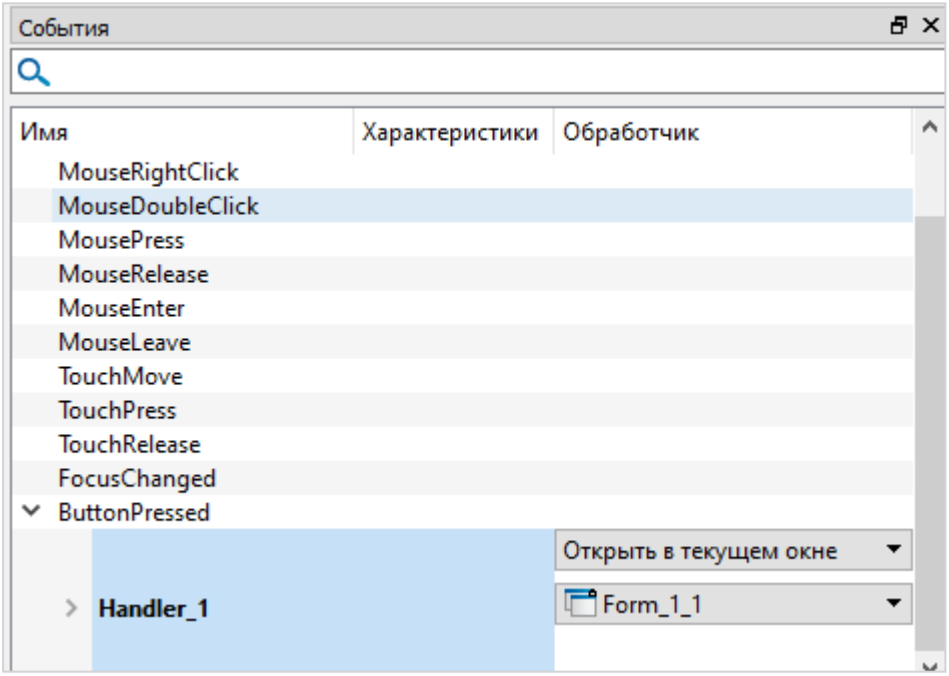
7. Работа с обработчиками событий

Чтобы реагировать на возникновение различных ситуаций у объектов существует понятие События. Таким образом, чтобы приложение выполняло некоторую работу в ответ на некое событие, разработчик должен определить реакцию на возникновение события - обработчик события (например, обработчик на событие двойного клика мыши по объекту MouseDoubleClick). Любое событие может иметь параметры, которые его характеризуют (например, для события двойного клика мыши по объекту, параметрами будут выступать локальные координаты X и Y, где был произведен этот клик). Все события выделенного объекта можно посмотреть в области **События**.

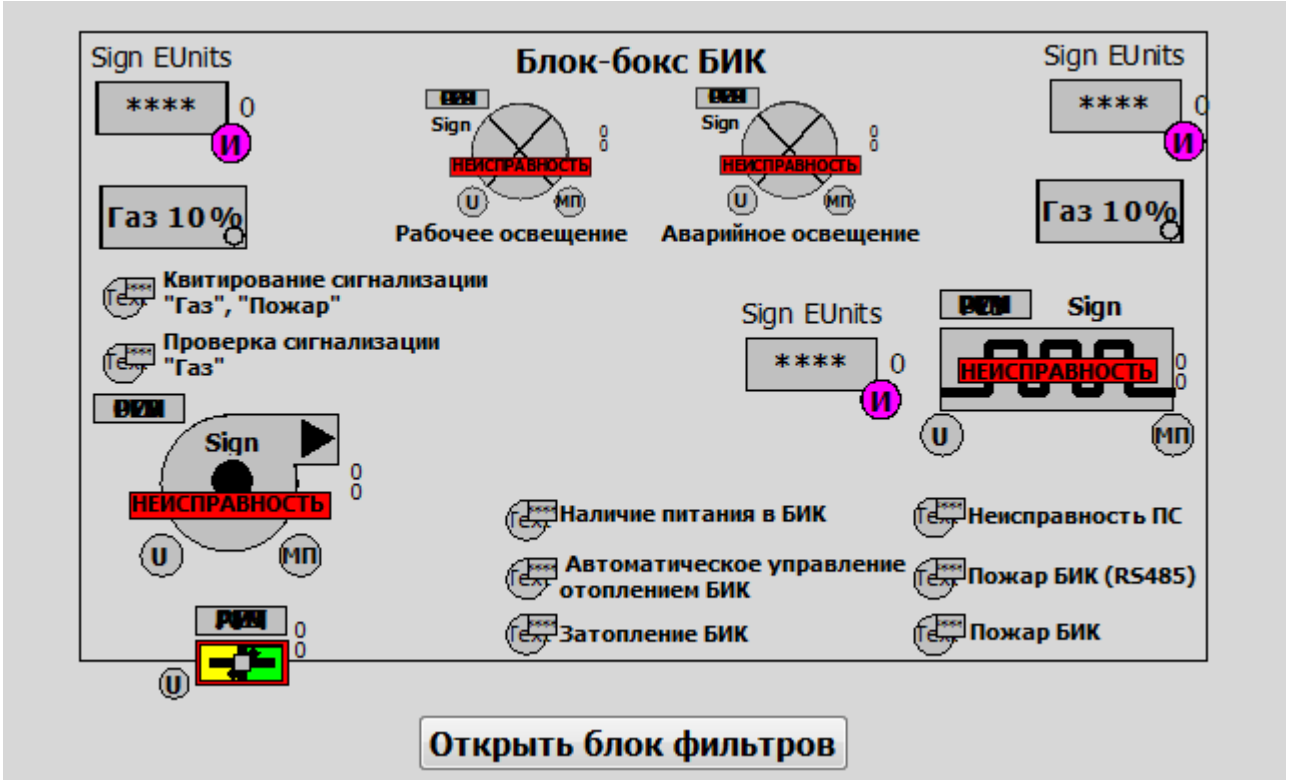
7.1. Открытие формы

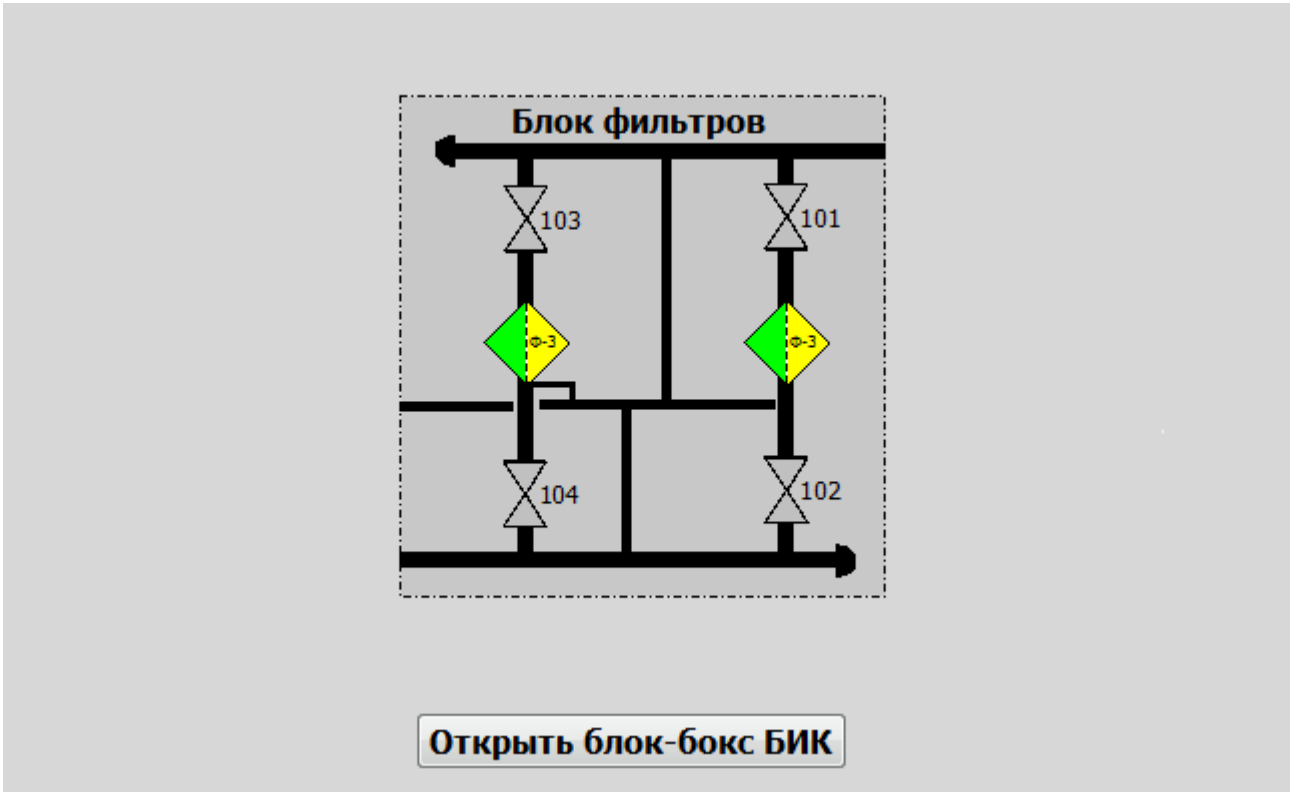
7.1.1. В текущем окне

Тип обработчика **«Открыть в текущем окне»** применяется для открытия экранной формы в текущем окне. На рисунке ниже показан обработчик события ButtonPressed для кнопки.



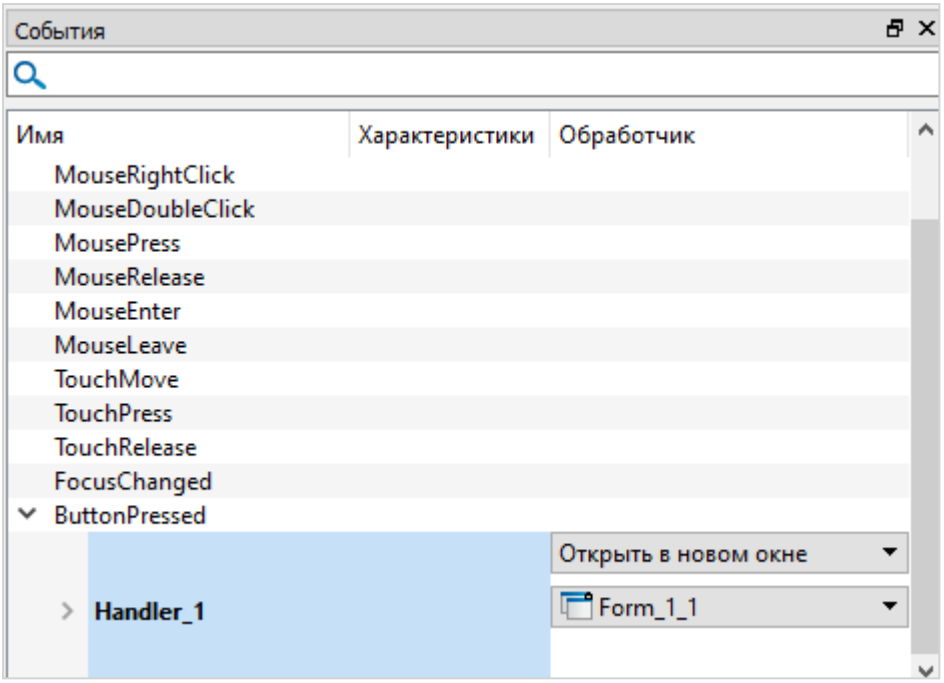
В результате, при нажатии на кнопку **«Открыть блок фильтров»**, в текущем окне откроется другая экранная форма со схемой блока фильтров.





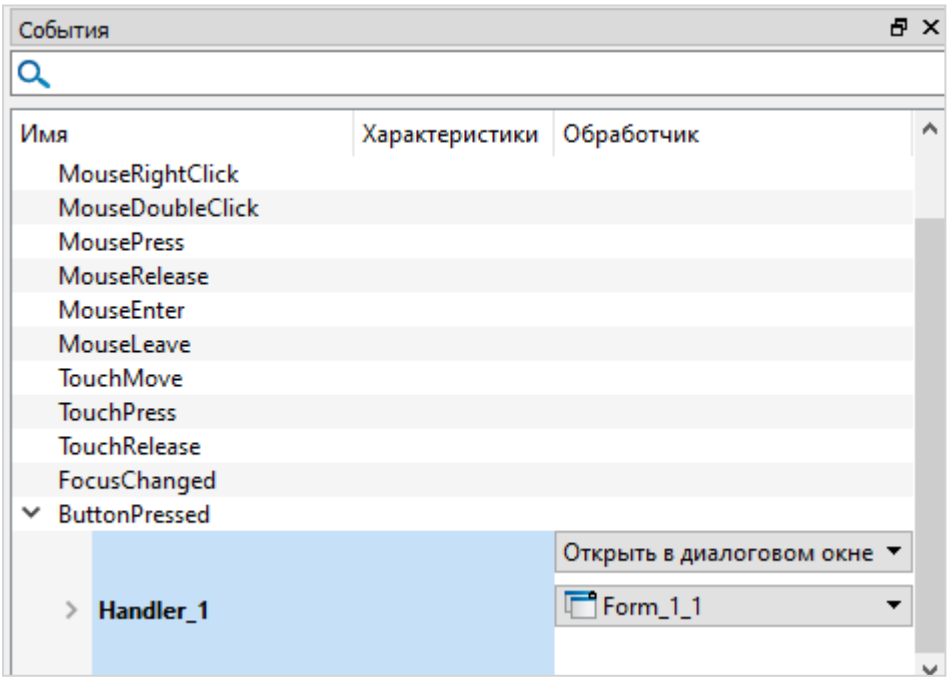
7.1.2. В новом окне

Тип обработчика **«Открыть в новом окне»** применяется для открытия экранной формы в новом окне. После открытия новой экранной формы, обе формы остаются активными, между ними можно переключаться. На рисунке ниже показан обработчик события ButtonPressed для кнопки.



7.1.3. В диалоговом окне

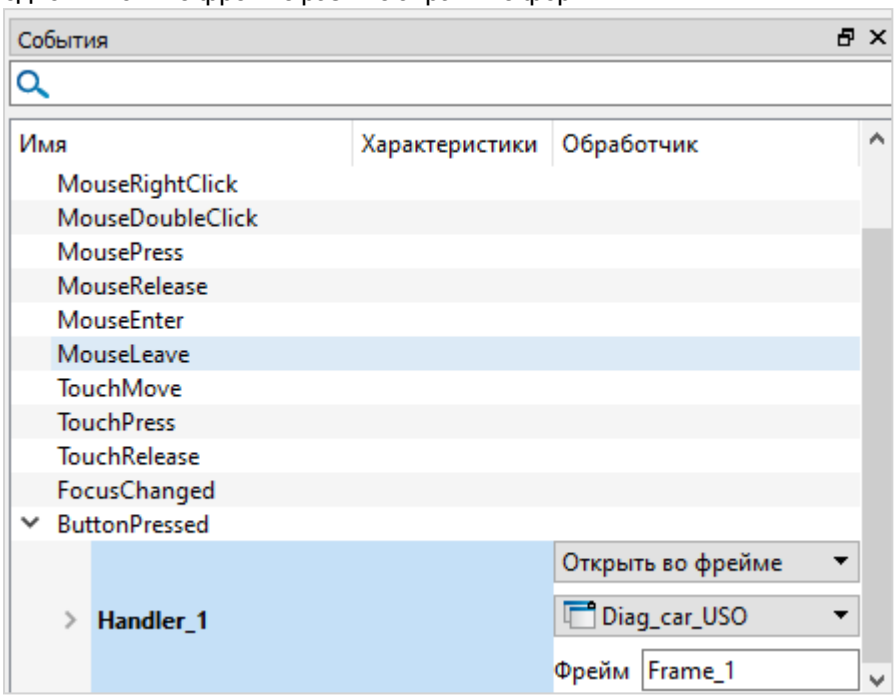
Тип обработчика «**Открыть в диалоговом окне**» применяется для открытия поверх основного окна другой экранной формы. Пока открыто диалоговое окно, у пользователя отсутствует возможность взаимодействовать с родительским окном. На рисунке ниже показан обработчик события ButtonPressed для кнопки.

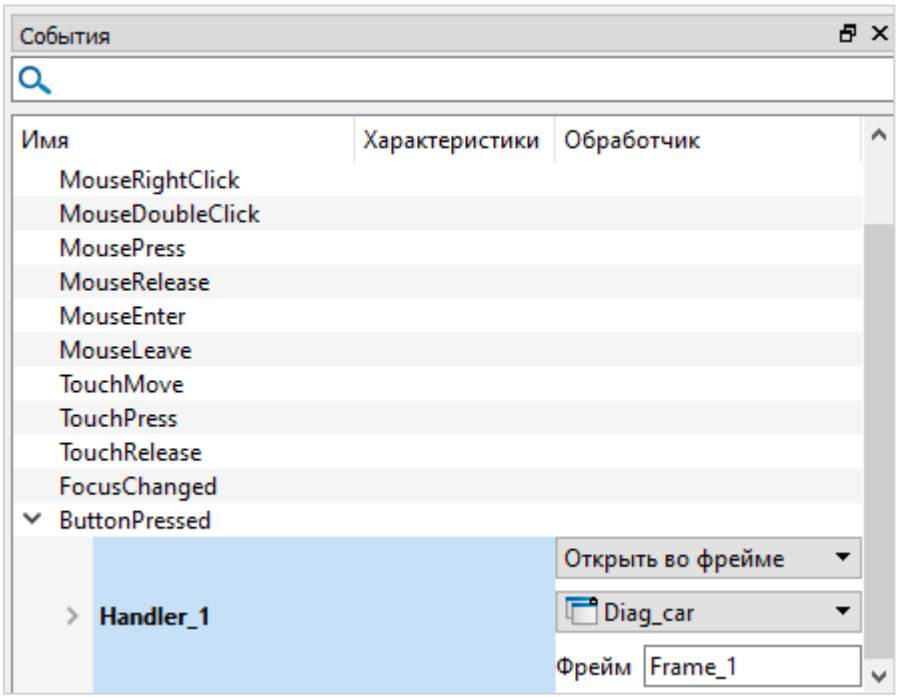


7.1.4. Во фрейме

Тип обработчика «**Открыть во фрейме**» применяется для динамического открытия экранных форм внутри фреймовой области.

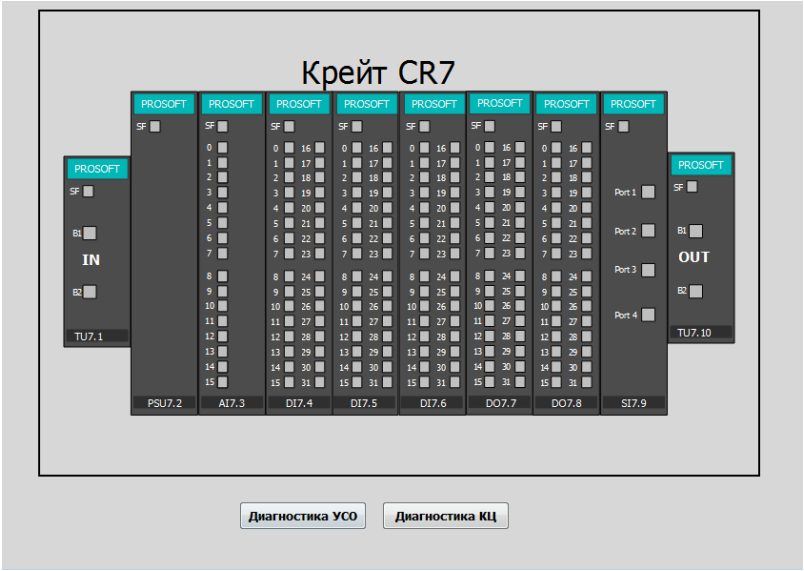
Чтобы открыть форму во фрейме, следует выбрать тип обработчика событий **«Открыть во фрейме»**. В обработчике укажите экранную форму и имя фрейма, в котором откроется выбранная форма. На рисунках ниже показаны обработчики событий кнопок **Диагностика УСО** и **Диагностика КЦ**, которые запускают во в одном и том же фрейме разные экранные формы.

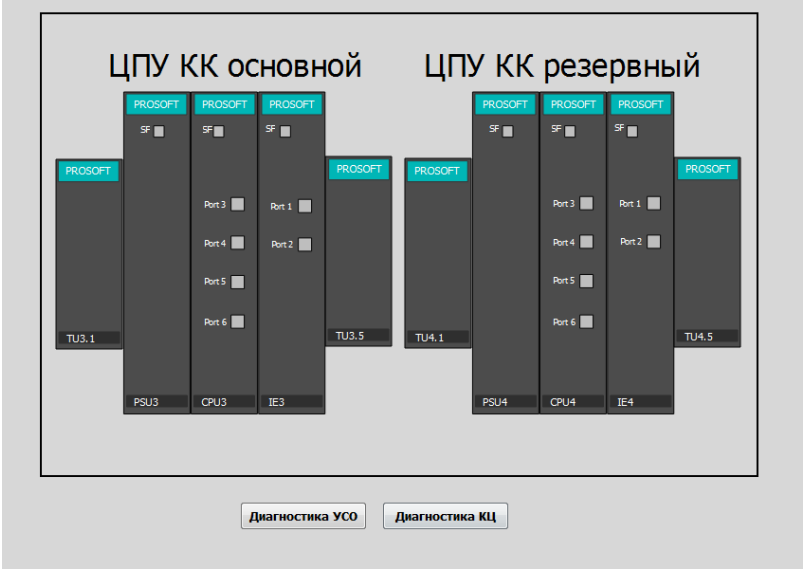




При запуске в рантайме откроется экранная форма с пустой фреймовой областью и двумя кнопками.

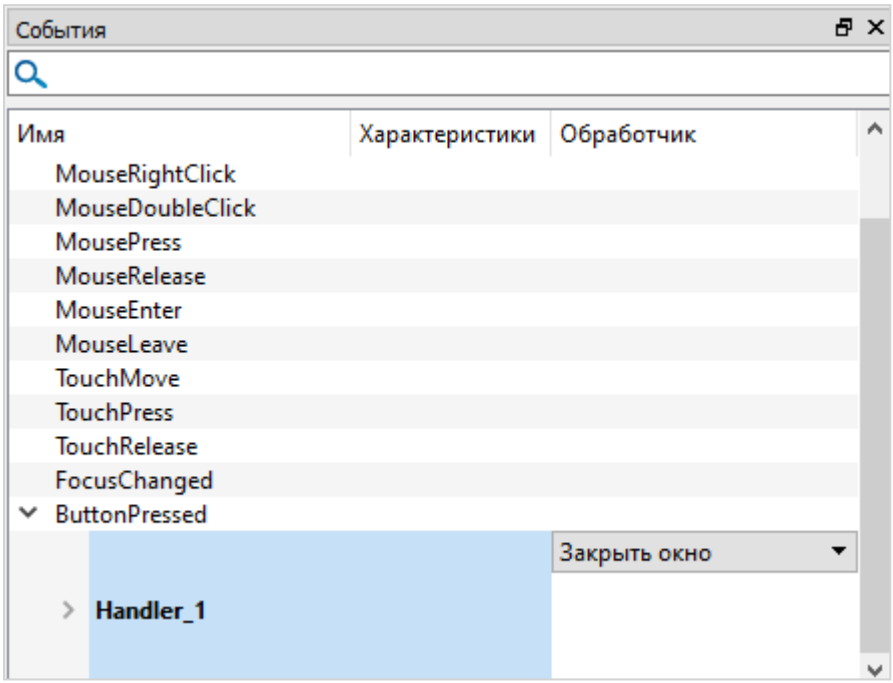
При нажатии на кнопку **Диагностика УСО** во фрейме откроется форма с диагностикой УСО. При нажатии на кнопку **Диагностика КЦ** во фрейме откроется форма с диагностикой КЦ.





7.2. Закрытие окна

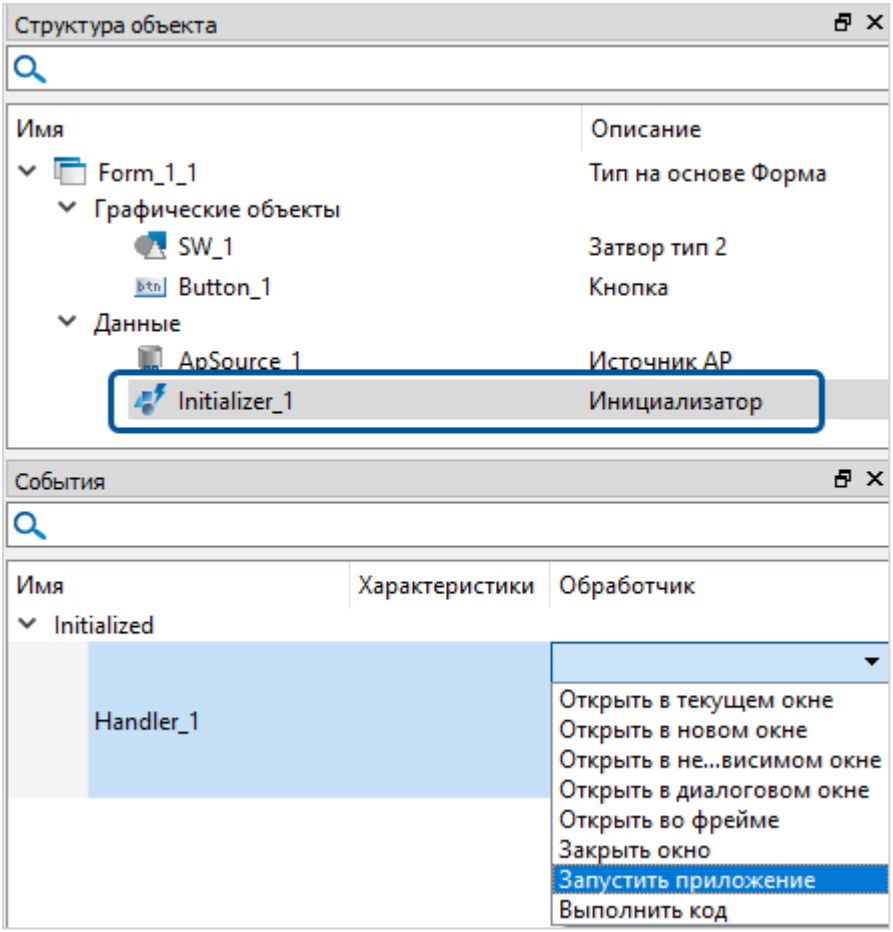
Тип обработчика **«Закрыть окно»** применяется для закрытия текущей экранной формы. На рисунке ниже показан обработчик события ButtonPressed для кнопки.



7.3. Событие при создании объекта

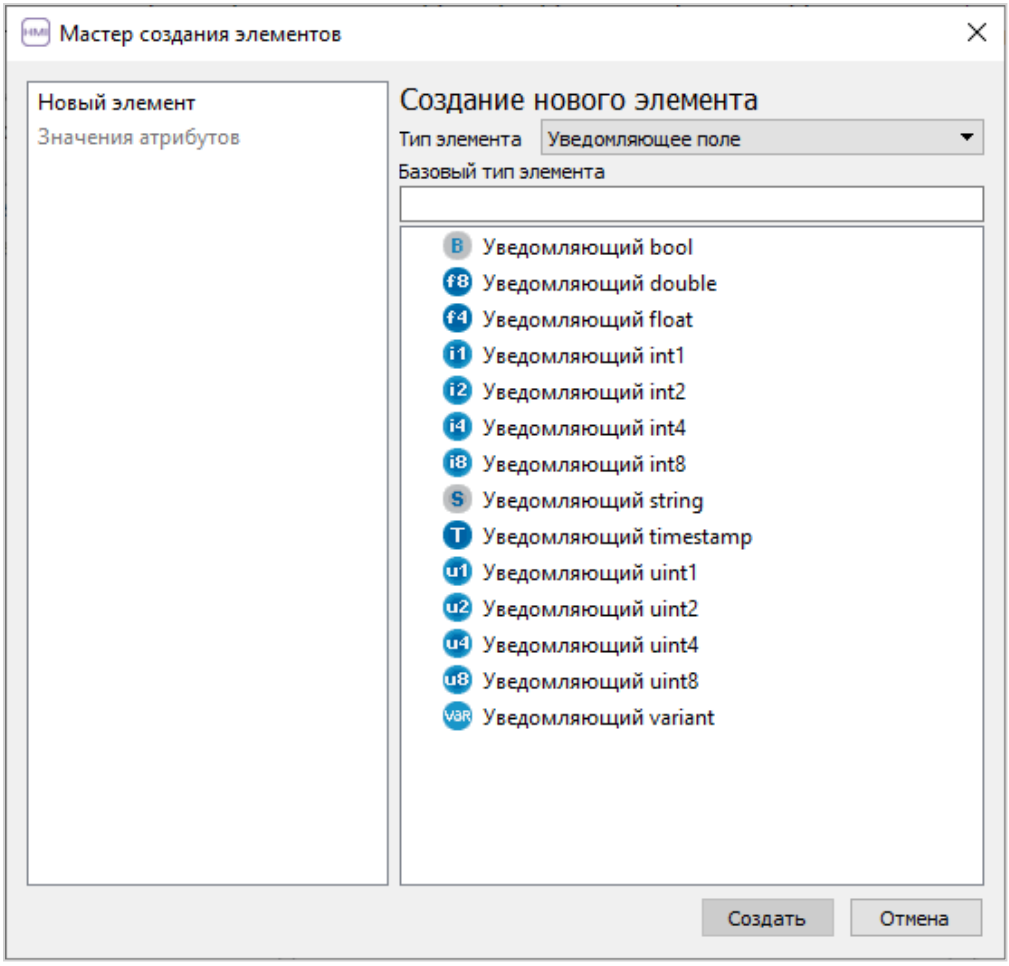
Чтобы определить события, которые будут происходить сразу после создания объекта или экранной формы в режиме исполнения, добавьте компонент **Инициализатор** в объект или на экранную форму. Компонент невизуальный и виден только в области **Структура объекта**.

После добавления **Инициализатора**, определите любой тип обработчика события (стр. 85).

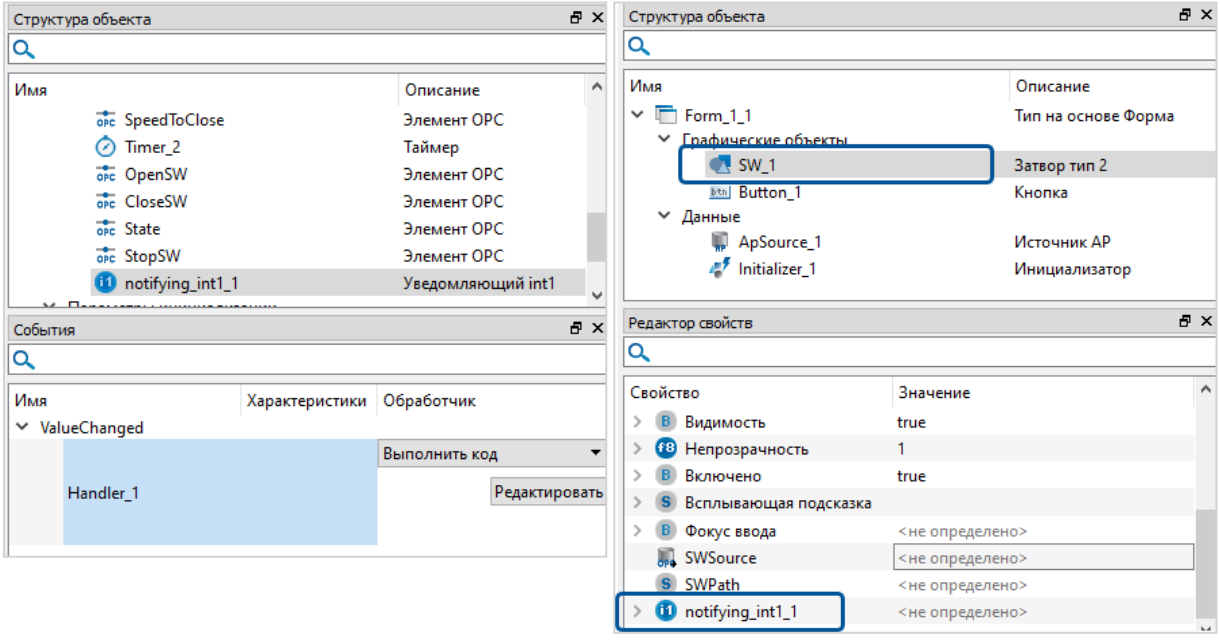


7.4. Событие при изменении свойства объекта

Чтобы каждое изменение свойства объекта приводило к возникновению события (стр. 85), применяйте уведомляющие поля. Уведомляющие поля удобны, когда требуется оперативная реакция на любое изменение важного параметра. Чтобы создать уведомляющее поле для объекта, выберите команду контекстного меню в Структуре объекта **Создать** и воспользуйтесь мастером создания элементов.



Уведомляющее поле выбранного типа добавится в Структуру объекта (группа **Данные**) и в список свойств объекта. Чтобы определить событие, которое будет срабатывать каждый раз при изменении уведомляющего поля, создайте обработчик для события типа ValueChanged.

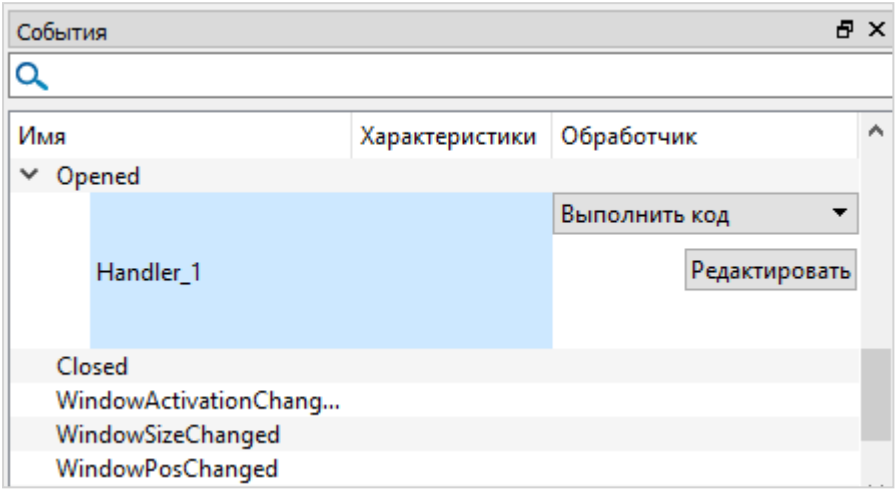


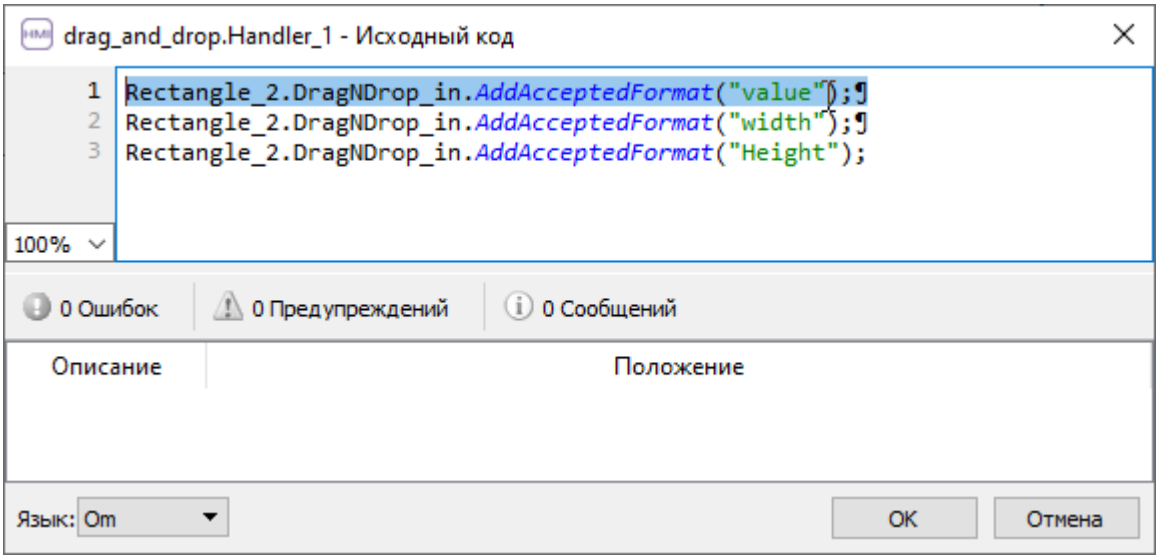
Параметры события ValueChanged:

Параметр	Описание
value	Новое значение уведомляющего поля.

7.5. Выполнение скриптов

Чтобы написать скрипт, выполняемый при наступлении какого-либо события, используйте обработчик типа **Выполнить код**. Для открытия редактора кода нажмите кнопку **Редактировать**.

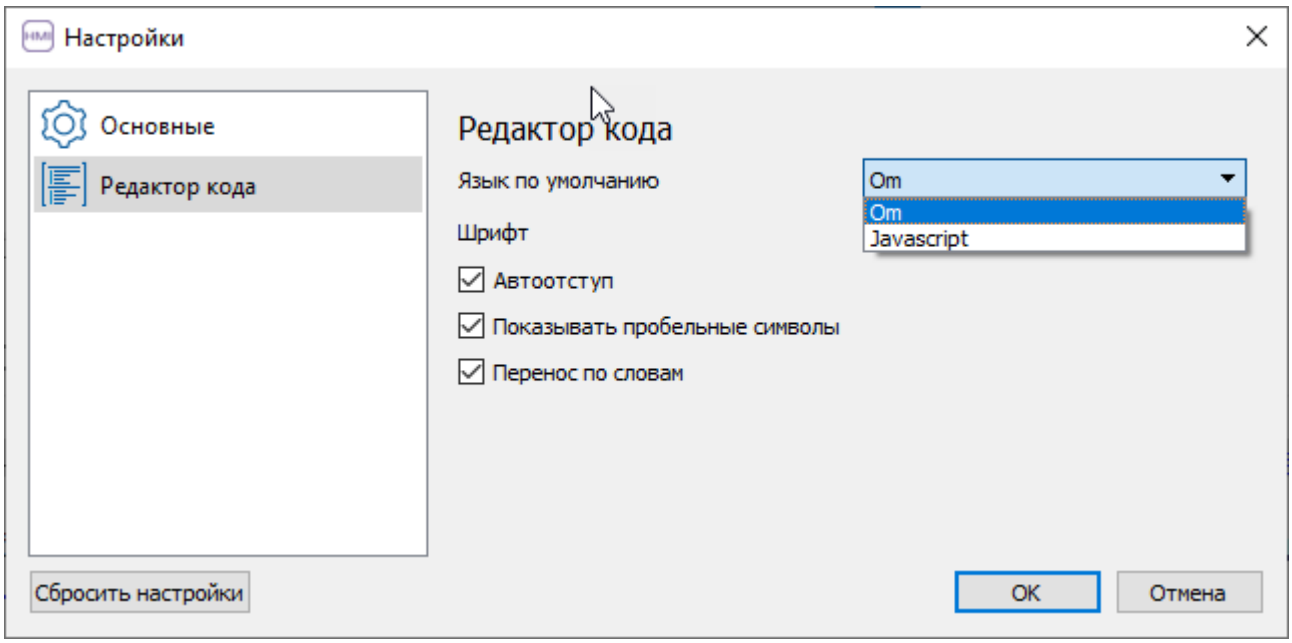




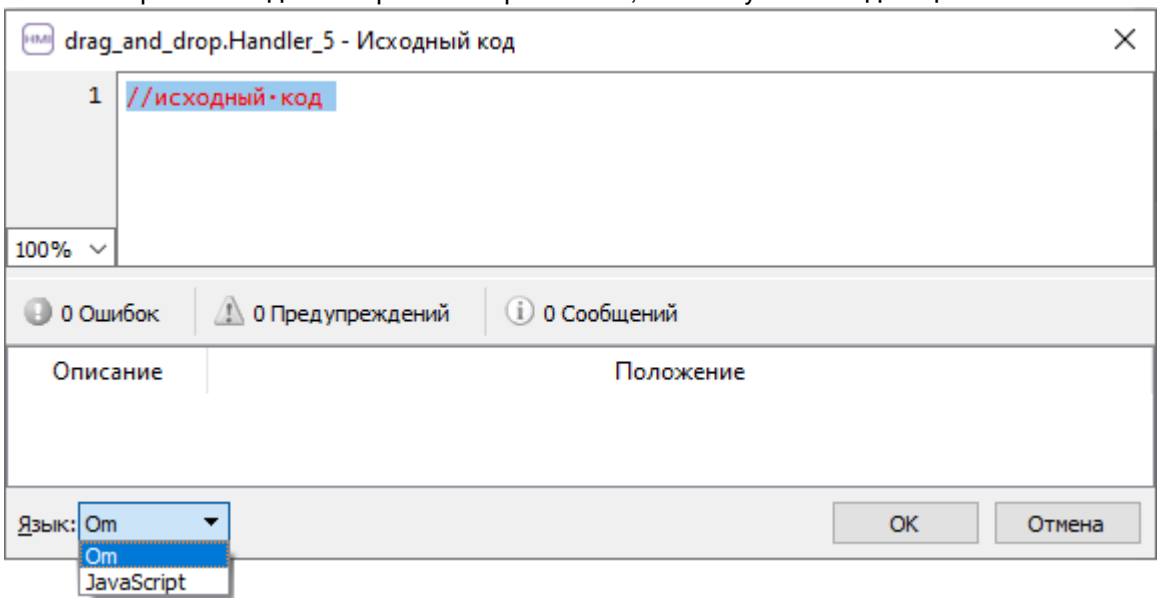
7.5.1. Выбор языка

В SePlatform.HMI доступны два языка для выполнения пользовательских скриптов:SePlatform.Omи JavaScript.

Чтобы выбрать язык, по умолчанию используемый в каждом редакторе исходного кода, перейдите в меню **Файл**, выберите **Настройки** и укажите язык на вкладке **Редактор кода**.



Чтобы выбрать язык для конкретного обработчика, используйте выпадающий список окна **Исходный код**.



7.5.1.1. SePlatform.Om

SePlatform.Om является единым скриптовым языком для различных программных продуктов Систэм Платформ:

- сервер ввода/вывода SePlatform.Data Server (сигнальная модель данных);
- среда разработки проектов автоматизации SePlatform.Development Studio (объектная модель данных);
- среда разработки и исполнения визуальной части проектов автоматизации SePlatform.HMI (объектная модель данных).

Для подробного ознакомления с возможностями языка и его синтаксисом см. документ Язык SePlatform.Om. Общее описание.

В SePlatform.HMI язык применяется для исполнения формул, обработчиков функций (стр. 99), обработчиков событий (стр. 85) и т.д.

ПРИМЕЧАНИЕ

Если вам недостаточно возможностей языка SePlatform.Om или использование языка вызвало какие-либо трудности, следует переключиться на язык JavaScript (стр. 93).

7.5.1.2. JavaScript

Язык JavaScript применяется для расширения стандартных возможностей языкаSePlatform.Om. Большинство возможностей языка JavaScript (в соответствии с ECMAScript Language Specification) доступно для применения в SePlatform.HMI. Исключение составляют браузерно-ориентированные возможности языка. Используемые в SePlatform.HMI встроенные функции JavaScript, методы встроенных объектов или собственные функции (стр. 99) могут возвращать значение, обращаться к входным аргументам в функции, а также вызывать другие функции.

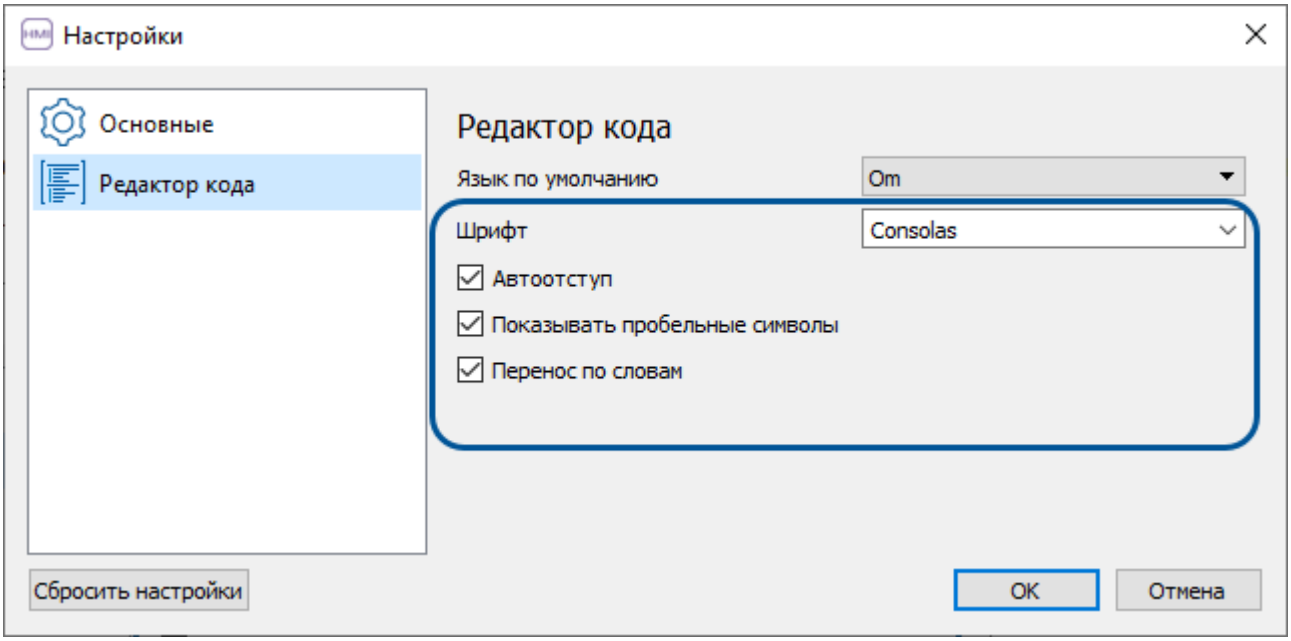
ОБРАТИТЕ ВНИМАНИЕ

Следует помнить, что из-за технологии динамической компиляции, скрипты JavaScript не проверяются на наличие ошибок в момент запуска режима исполнения. Все ошибки и предупреждения, возникающие в ходе исполнения скриптов JavaScript, будут логироваться в области **Журнал времени исполнения**.

7.5.2. Улучшение внешнего вида кода

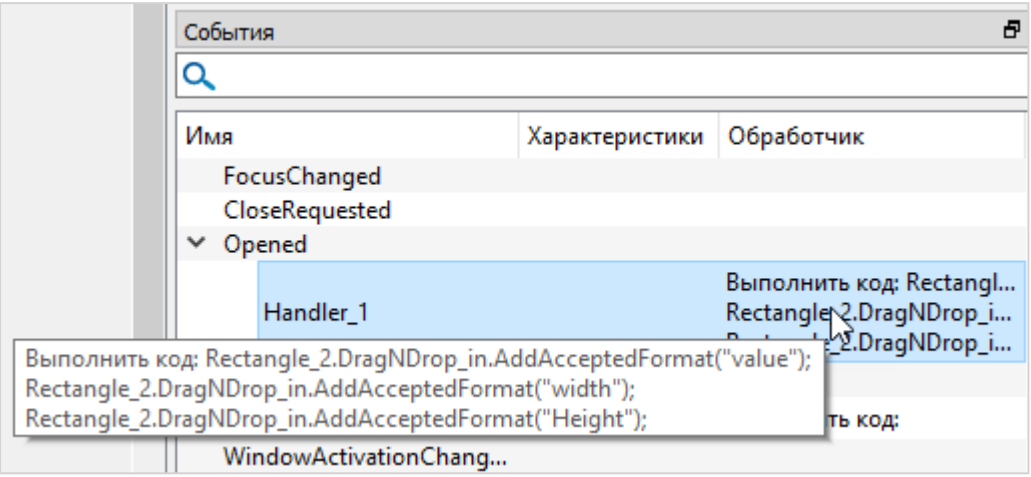
При написании кода используйте горячие клавиши **«Tab»**, **«Shift»+«Tab»** для ввода табуляции.

Чтобы настроить внешний вид кода по умолчанию, перейдите в меню **Файл**, выберите **Настройки** и включите нужные опции на вкладке **Редактор кода**.



ПРИМЕЧАНИЕ

Чтобы посмотреть полный текст кода, не открывая редактор обработчика, наведите на код курсор мыши.



7.5.3. Относительная адресация в SePlatform.Om

Чтобы обращаться к объектам на различных уровнях иерархии , используйте при написании скриптов специальные ключевые слова:

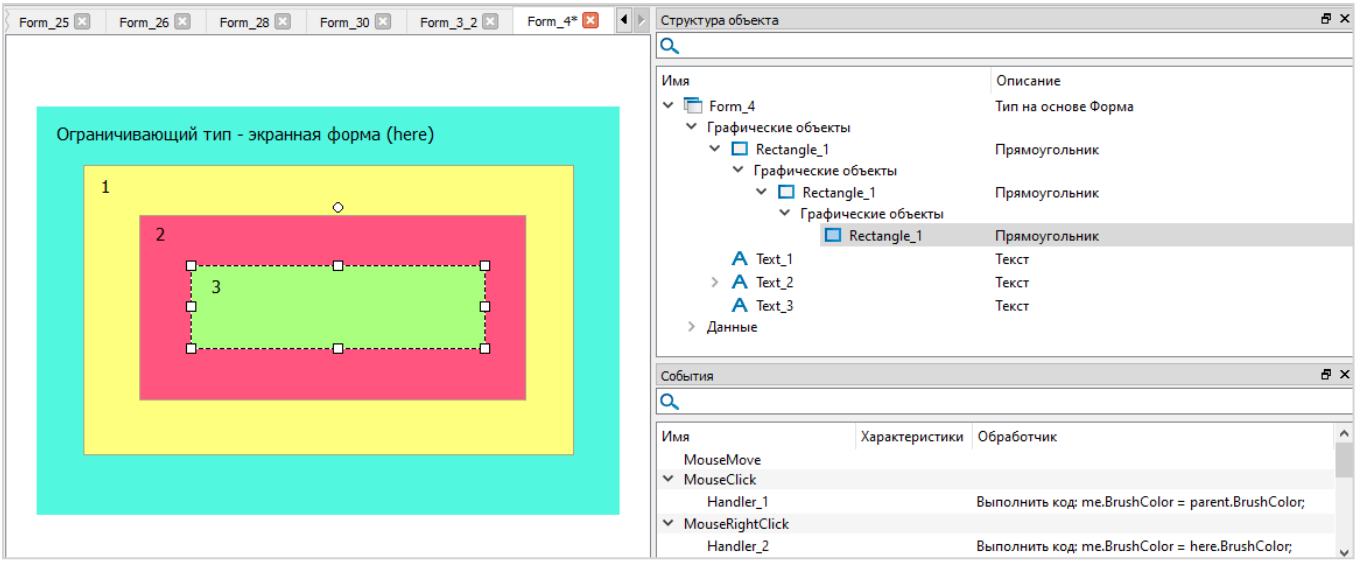
- me или this - обращение к текущему объекту;
- parent - обращение к родителю текущего объекта;
- here - обращение к ограничивающему типу (корню иерархии в рамках типа).

7.5.3.1. Пример относительной адресации

Для демонстрации относительной адресации в примере рассмотрена иерархия объектов экранной формы. Созданы 3 прямоугольника, которые вложены друг в друга. Требуется, чтобы по левому щелчку мыши на любом прямоугольнике происходило его окрашивание в цвет родительского объекта, а по правому щелчку мыши - происходило его окрашивание в цвет ограничивающего типа.

ПРИМЕЧАНИЕ

В данном примере ограничивающий тип - сама экранная форма (бирюзовый цвет).



Для каждого прямоугольника экранной формы создайте обработчик MouseClick, который выполняет следующий код:

```
//окрасить текущий объект в цвет родительского объекта
me.BrushColor = parent.BrushColor;
```

Для каждого прямоугольника экранной формы создайте обработчик MouseRightClick, который выполняет следующий код:

```
//окрасить текущий объект в цвет ограничивающего типа (цвет самой экранной формы)
me.BrushColor = here.BrushColor;
```

7.5.4. Параметры события

Ниже приведен перечень типовых событий, которые встречаются у большинства объектов для построения графических примитивов или пользовательского интерфейса (**Линия**, **Прямоугольник**, **Эллипс** и т.д.).

Вышеперечисленные события имеют параметры с доступом через структуру event:

Параметр	Тип	Описание
X	double	Локальная координата объекта по оси X, в которой возникло событие.
Y	double	Локальная координата объекта по оси Y, в которой возникло событие.
Buttons	int4	Код кнопок мыши, которые были нажаты при возникновении события.
KeyboardModifiers	int4	Коды клавиш клавиатуры, которые были зажаты при возникновении события.

ПРИМЕР

```
//Разместить кнопку в месте срабатывания события
Button.X = event.X;
Button.Y = event.Y;
//Записать в текстовое поле код кнопок мыши, которые были зажаты при возникновении
события
TextEdit_1.Text = Str.ToString(event.KeyboardModifiers);
//Записать в текстовое поле код клавиш клавиатуры,
// которые были нажаты при возникновени события
TextEdit_2.Text = Str.ToString(event.Buttons);
```

8. Работа с функциями объектов

Функции - вызываемые процедуры, позволяющие влиять на состояние объектов или их свойств. Как правило, у функций есть набор входных параметров. К примеру, у объекта стандартной библиотеки компонентов **Выпадающий список** есть предустановленная (стр. 99) функция **RemoveItem(Index)**, предназначенная для удаления элемента списка.

В SePlatform.HMI можно определять свои собственные (стр. 99) функции для любых объектов, используя язык SePlatform.Om или JavaScript. Любая созданная функция может возвращать значение, обращаться к входным аргументам в функции, а также вызывать другие функции.

8.1. Предустановленные функции

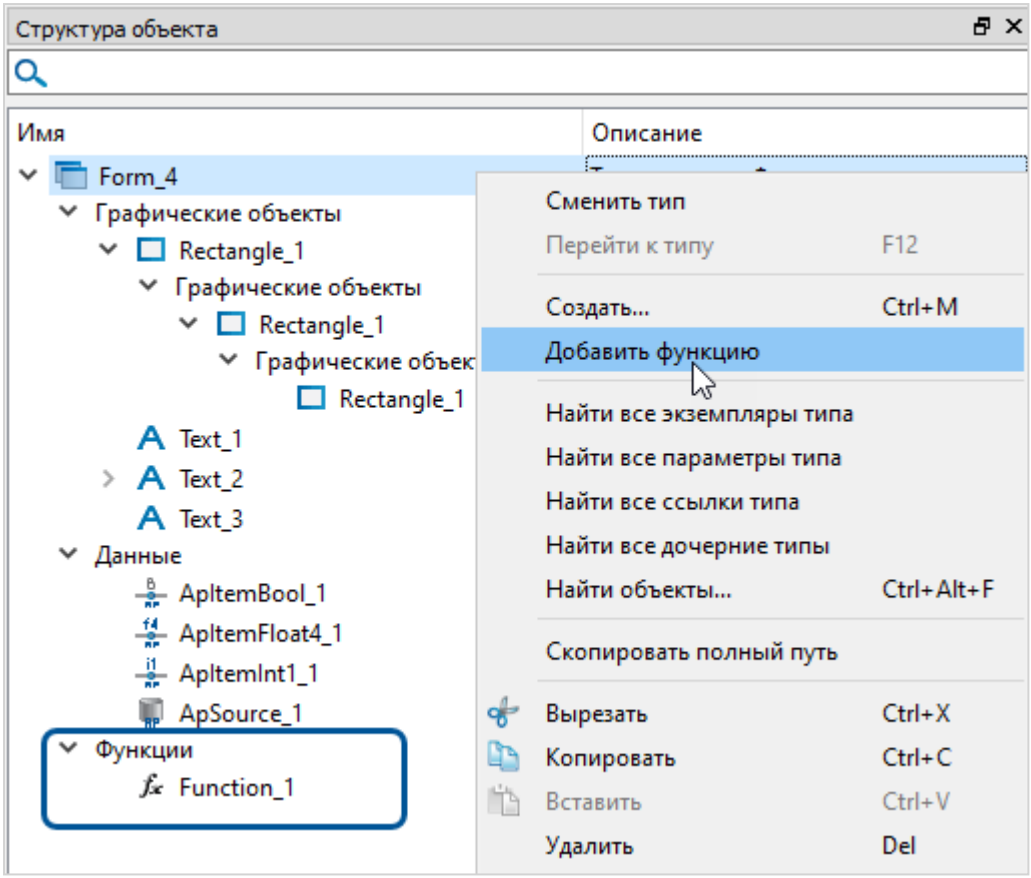
Некоторые объекты, созданные с помощью стандартной библиотеки компонентов SePlatform.HMI, имеют предустановленные функции. К примеру, выпадающий список, добавленный на экранную форму, будет иметь набор предустановленных функций для работы с ним:

- очистка списка - **Clear()**;
- добавление элемента в список - **AddItem(Text)**;
- удаление элемента из списка - **RemoveItem(Index)**;
- модификация элемента списка - **SetItem(ItemIndex,Text)** и т.д.

8.2. Создание собственных функций

Для экранных форм, собственных типов или объектов стандартной библиотеки компонентов можно определять собственные функции.

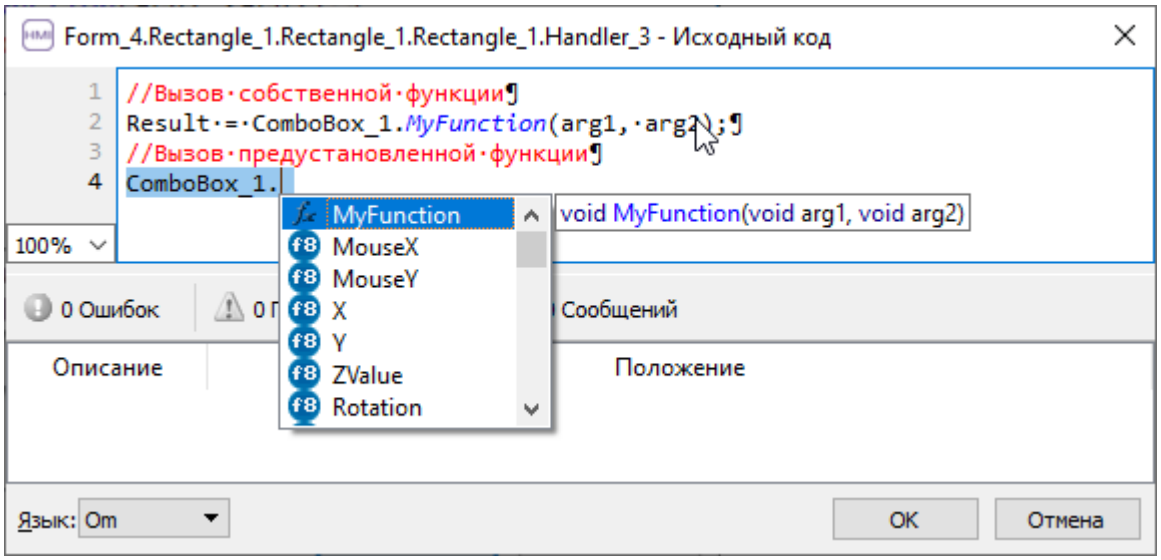
Чтобы создать собственную функцию для какого-либо элемента, вызовите контекстное меню в области **Структура объекта** и выберите. Собственные функции появляются в разделе **Функции** области **Структура объекта**.



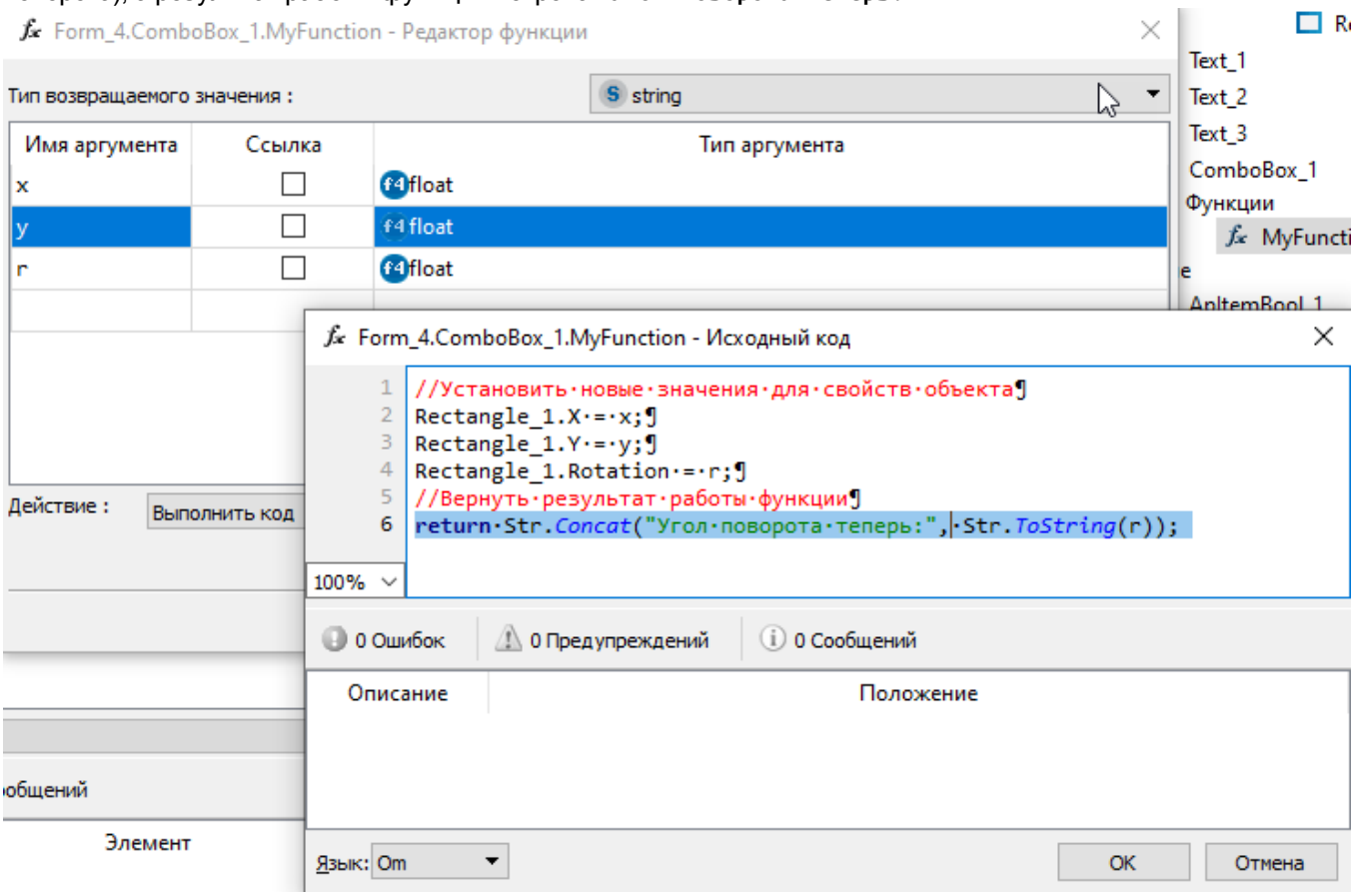
В окне **Редактор функции** определите входные аргументы (имена аргументов и их типы). Также укажите тип действия, которое будет выполнять функция (типы действий аналогичны обработчикам событий (стр. 85)). При необходимости, можно сменить тип возвращаемого значения или оставить его пустым (void).

8.3. Вызов функций и возврат значений

Чтобы обращаться к функциям объекта из скриптов и формул, используйте точку в качестве разделителя между именем объекта и именем функции. В процессе написания скриптов или формул возможные варианты функций объекта можно выбирать из выпадающего списка, который появляется после ввода точки. Просмотреть назначение функции и входные параметры можно во всплывающей подсказке.

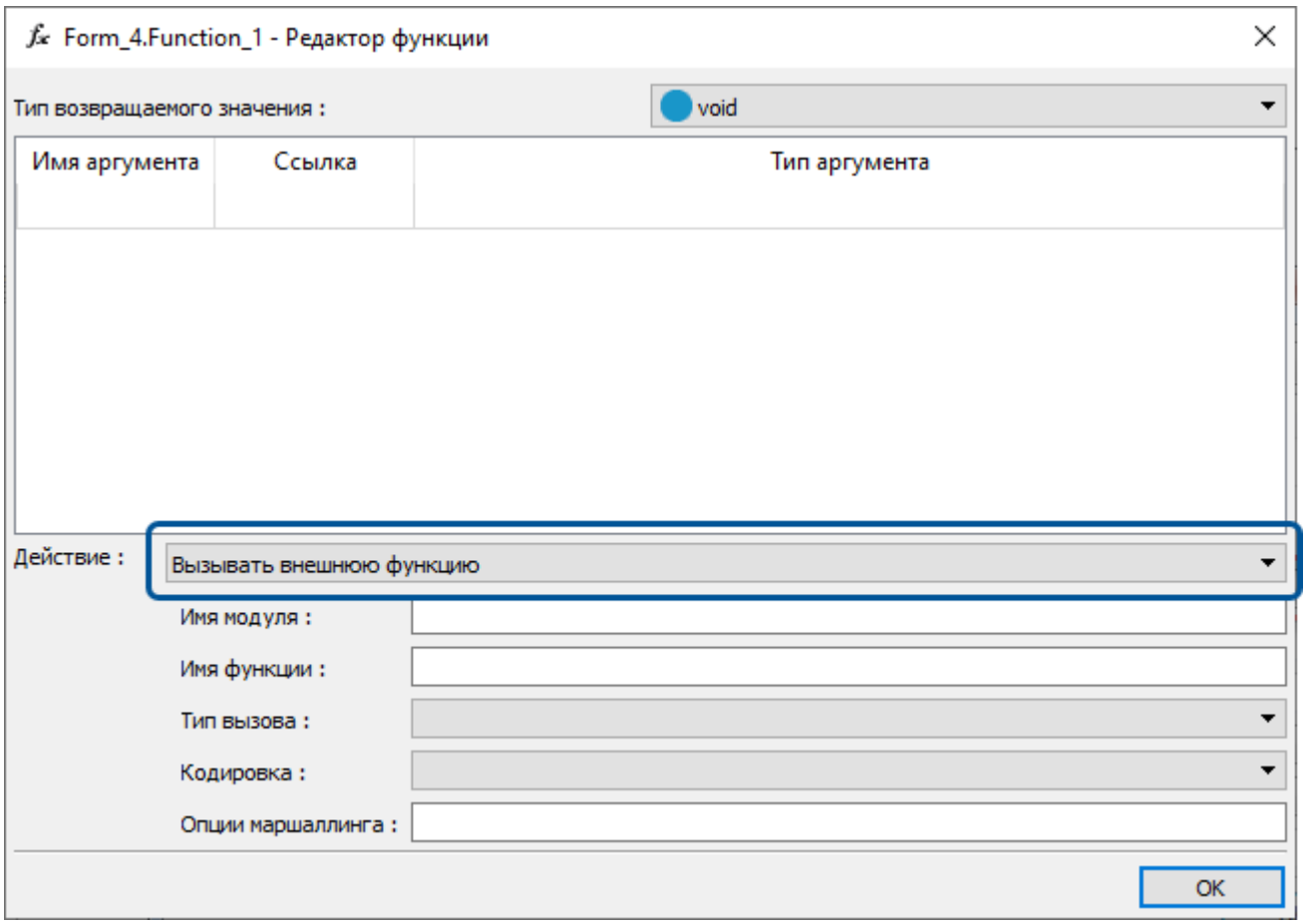


Если функция возвращает какой-либо результат, то для возврата значения используйте оператор return. В примере ниже показана функция, которая изменяет свойства прямоугольника (ширину, высоту и угол поворота), а результат работы функции - строка **«Угол поворота теперь: N»**.



8.4. Вызов внешних функций

Чтобы вызывать функции, определенные в сторонних динамических библиотеках (DLL-файлах), используйте тип действия **«Вызывать внешнюю функцию»** в **Редакторе функций**.



Определите параметры вызова внешней функции:

Параметр	Описание
Имя модуля	Абсолютный путь до DLL-файла. Если DLL- файл расположен в корне каталога (стр. 11) с установленным SePlatform.HMI, то достаточно указать его имя без пути
Имя функции	Имя вызываемой функции, которая определена в файле динамической библиотеки
Тип вызова	Способ вызова функции CDecl или StdCall (Win API). Зависит от спецификации вызываемой внешней функции
Кодировка	Кодировка символов: › **«UTF8»** › **«UTF16»** › **«ANSI»**
Опции маршалинга	Настройки направления работы аргументов (входной, выходной или с применением буфера). Все аргументы по умолчанию - входные. Если аргумент выходной, то это нужно указать в опциях маршалинга: **«["out"]»**, если входной/выходной - **«["in", "out"]»**. Если выходной аргумент типа string, то опции маршалинга: **«["out", "buffer"]»**.

Пример

Разберем детально процесс вызова внешней функции ExtLibCalcNoErrorDesc, которая определена в файле `extlib.dll`.

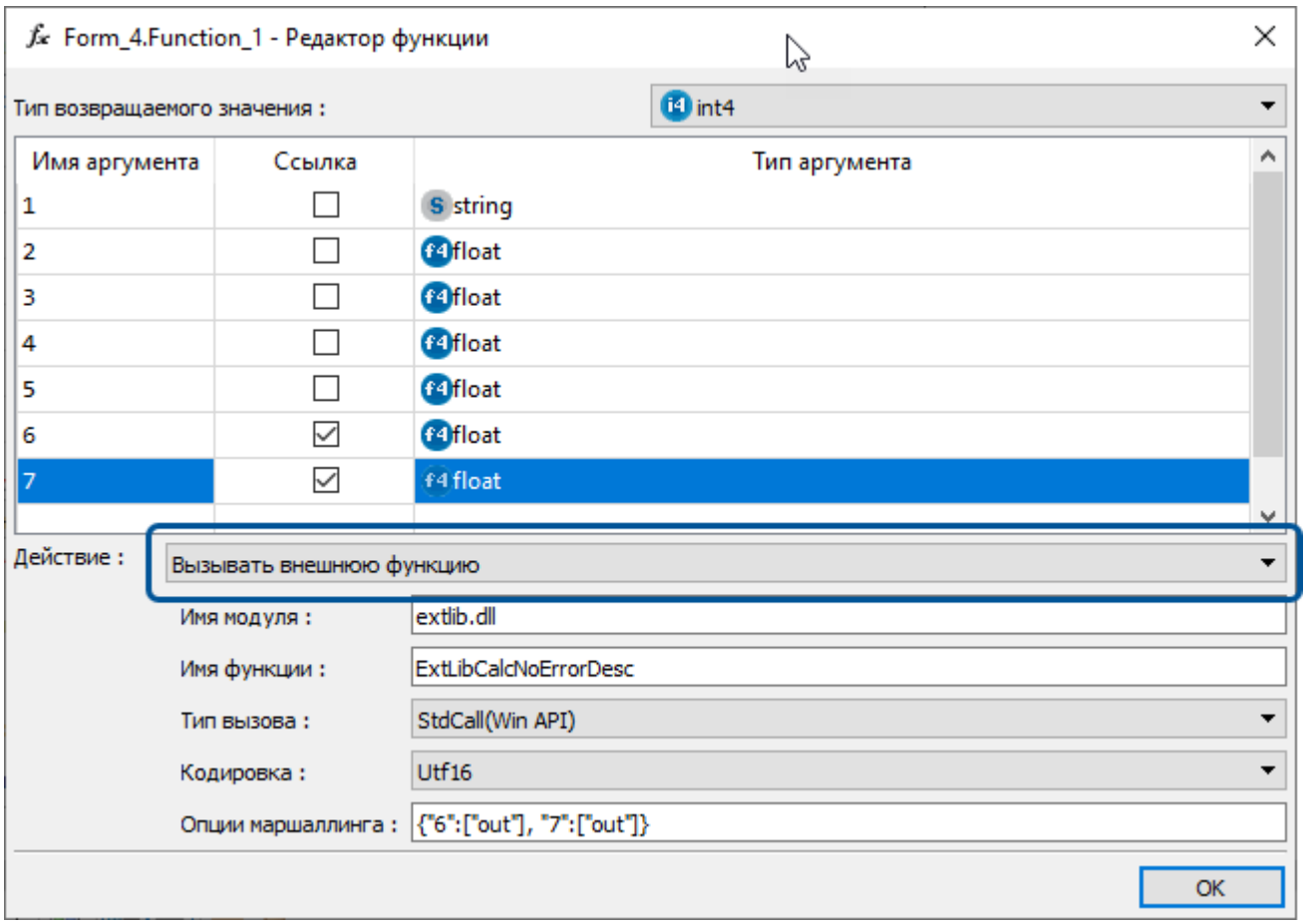
Ниже показан фрагмент исходного кода динамической библиотеки:

```
std::int32_t __stdcall ExtLibCalcNoErrorDesc(
  wchar_t const *inputString,
  double a1,
  double x1,
  double a2,
  double x2,
  double *y1,
  double *y2)
{
  try
  {
    //Проверяем входные параметры
    If( !inputString ) throw Error( L"Не указана входная строка" );
    // Считаем выражения
    *y1 = a1 * x1 + a2 * x2;
    *y2 = ( 1 / a1 ) * x1 - ( 1 / a2 ) * x2;
    // Всё прошло успешно, вернуть 1
    return 1;
  }
  // Ловим ошибку и возвращаем 0
  catch( Error const &e )
  {
    return 0;
  }
}
```

По исходному коду видно, что функция принимает значения 7 входных параметров (2 из них передаются по ссылке), выполняет с ними некоторые расчеты и возвращает **«1»** или **«0»**.

Создайте новую функцию (стр. 99)ExtLibCalcNoErrorDesc и настройте параметры:

- Тип возвращаемого значения выберите в соответствии с типом возвращаемого значения самой функции - это int4;
- Перечень входных аргументов настройте в соответствии с типами входных аргументов самой функции: первый аргумент строковый, все остальные - тип float8;
- Два последних аргумента функции являются ссылочными - отметьте их соответствующими флагами в колонке **Ссылка**;
- Выберите в выпадающем списке действие **«Вызывать внешнюю функцию»**;
- Укажите **Имя модуля** - название файла, из которого будет вызываться внешняя фукнция - **«extlib.dll»**;
- Укажите **Имя функции**, которую требуется вызывать - **ExtLibCalcNoErrorDesc**;
- Укажите **Тип вызова**. Для вызываемой функции - **«StdCall(Win API)»**;
- Укажите кодировку - **«Utf16»**;
- Укажите **Опции маршалинга** в формате: **«{"имя параметра": ["in","out","buffer"]}»**.



Ниже показан участок кода, вызывающий внешнюю функцию из скриптов:

```
//Готовим входные параметры для внешней функции
inp_str: string = "входная строка";
a1: double = 0.5;
a2: double = 0.75;
x1: double = 0.25;
x2: double = 0.01;
_y1: double = 0;
_y2: double = 0;
//Вызвать внешнюю функцию и присвоить результат выполнения переменной Result
Result: ExtLibCalcNoErrorDesc (inp_str, a1, x1, a2, x2, &_y1, &_y2);
//Обработка результатов выполнения функций
if(Result == 0)
  TextField.Text = "Произошла ошибка";
else
  TextField.Text = "Внешняя функция выполнила свою работу";
```

ПРИМЕЧАНИЕ

Передача параметров по ссылке выполняется через символ «&».

9. Встраивание ActiveX компонентов и внешних библиотек .NET

В проект SePlatform.HMI можно встроить библиотеки на основе .NET Framework и .NET Core.

ОБРАТИТЕ ВНИМАНИЕ

Библиотеки .NET Framework недоступны для использования в ОС Linux.

С помощью библиотек на основе .NET Framework и .NET Core можно добавить дополнительные компоненты в библиотеку компонентов и расширить стандартные возможности SePlatform.HMI внутри отдельного проекта.

ПРИМЕЧАНИЕ

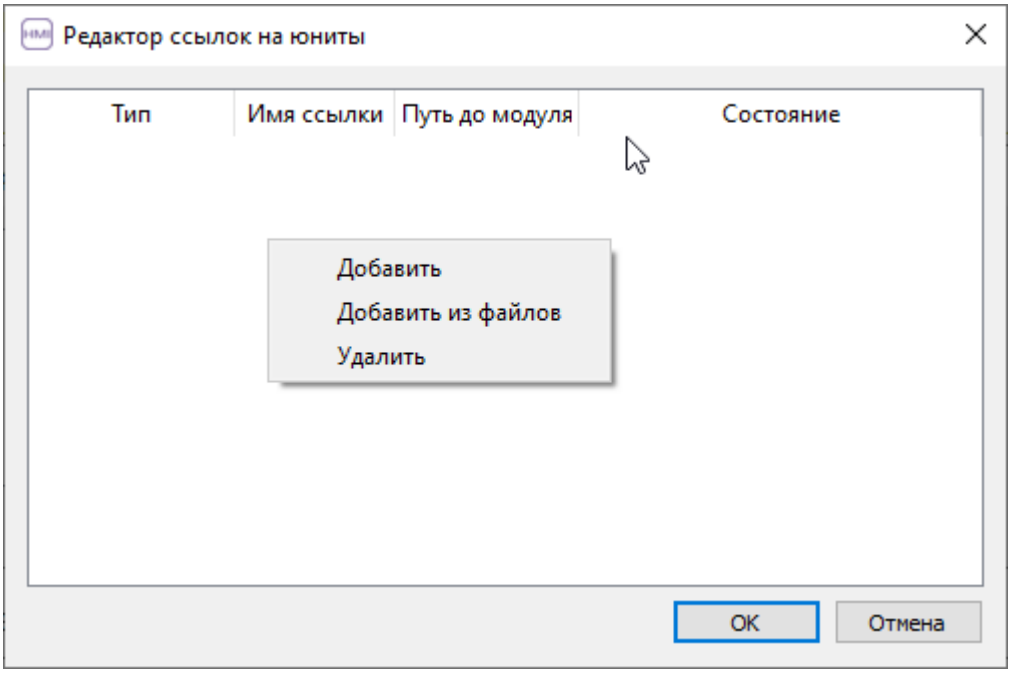
Для корректной работы библиотек .NET Core в ОС Linux установите пакеты среды выполнения .NET Core (см. ниже).

Чтобы добавить библиотеки в проект:

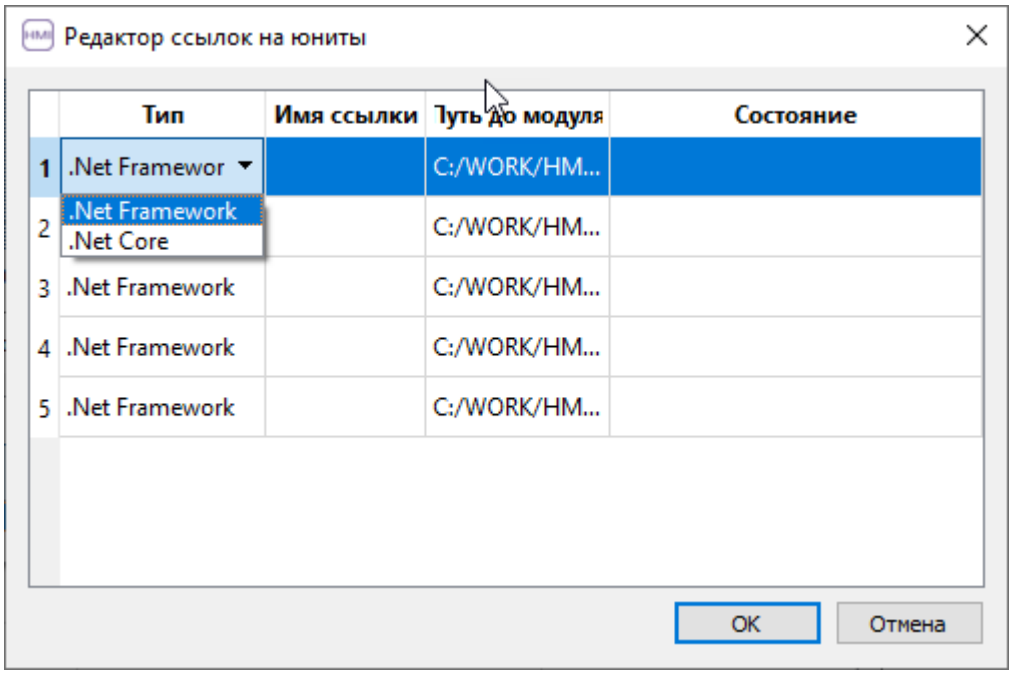
1. Выполните команду **Проект** → **Ссылки на юниты**. В открывшемся окне используйте команды контекстного меню для добавления или удаления библиотек.

ОБРАТИТЕ ВНИМАНИЕ

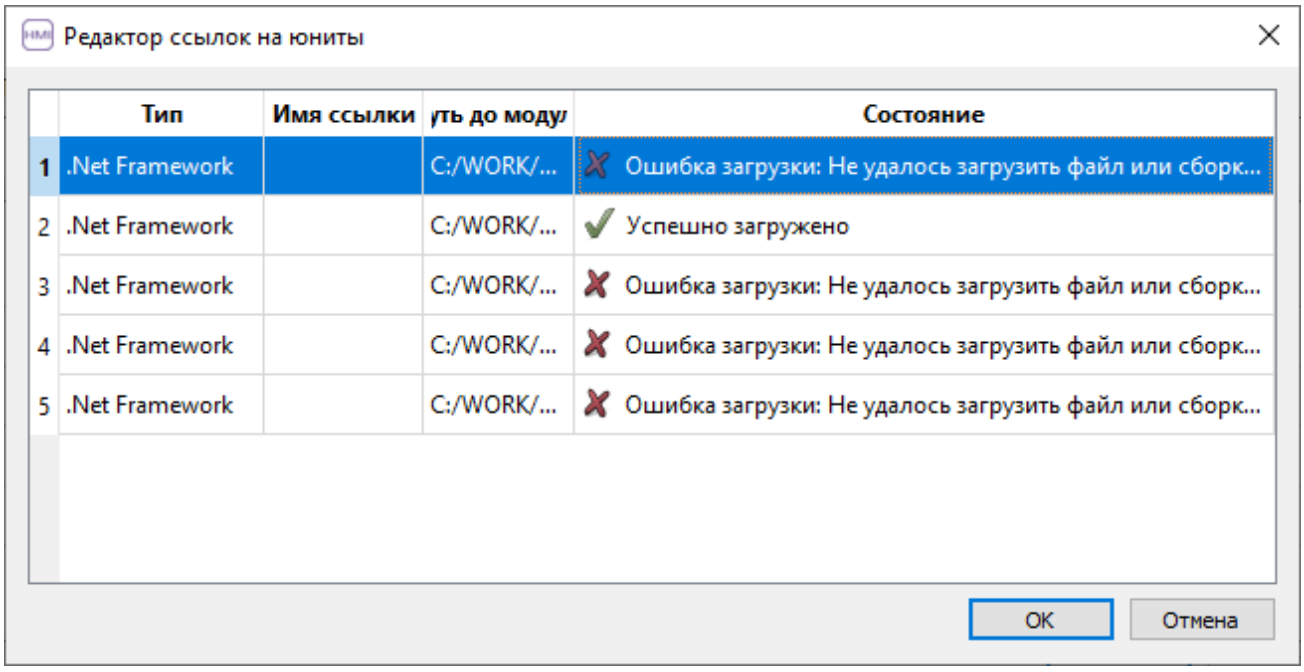
Компоненты .NET Framework недоступны при работе в ОС Linux.



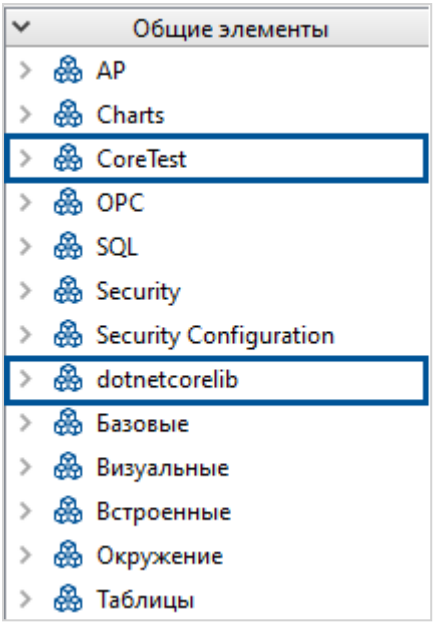
2. Укажите платформу для каждой библиотеки, в столбце **Тип**.



3. Чтобы просмотреть статус загрузки библиотеки, нажмите **ОК** в окне **Редактор ссылок на юниты** и повторно откройте окно.



Загруженные библиотеки добавляют в библиотеку компонентов SePlatform.HMI новые юниты с компонентами. Используйте новые компоненты аналогично встроенным компонентам SePlatform.HMI.



9.1. Установка пакетов среды выполнения .NET Core

ПРИМЕЧАНИЕ

Пакеты установки .Net Core не поставляются вместе с комплектом установочных пакетов Систэм Платформ.

Подробнее установка с использованием Internet описана здесь:
https://wiki.astralinux.ru/pages/viewpage.action?pageId=41192241.

Чтобы установить .Net Core без использования Internet, выполните:

1. Скачайте установочные пакеты командами:

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg - -dearmor
>microsoft.asc.gpg
wget -q https://packages.microsoft.com/config/debian/9/prod.list
```

2. Зарегистрируйте скачанные пакеты на компьютере и ограничьте к ним доступ командами:

```
sudo cp microsoft.asc.gpg /etc/apt/trusted.gpg.d/
sudo cp prod.list /etc/apt/sources.list.d/microsoft-prod.list
sudo chown root:root /etc/apt/trusted.gpg.d/microsoft.asc.gpg
sudo chown root:root /etc/apt/sources.list.d/microsoft-prod.list
```

3. Обновите репозитории и установите пакеты командами:

```
sudo apt-get update
sudo apt-get install dotnet-runtime-3.1
```

ПРИМЕЧАНИЕ

В текущем документе описана работа с dotnet runtime 3.1. Уточнить актуальную версию можно по ссылке: https://docs.microsoft.com/ru-ru/dotnet/core/install/linux-package-manager-debian9.

10. Формирование таблиц

Чтобы получить возможность:

- помещать данные, полученные от источника по TCP, в таблицу
- формировать собственные таблицы данных

следует подключить внешнюю библиотеку `seplatform.hmi.tables` (для ОС Windows) и `libseplatform.hmi.tables` (для ОС Linux).

Чтобы подключить библиотеки:

ОС Windows

Разместите файл библиотеки в папке установки SePlatform.HMI.

ОС Linux

Скопируйте библиотеку в папку SePlatform.HMI командой

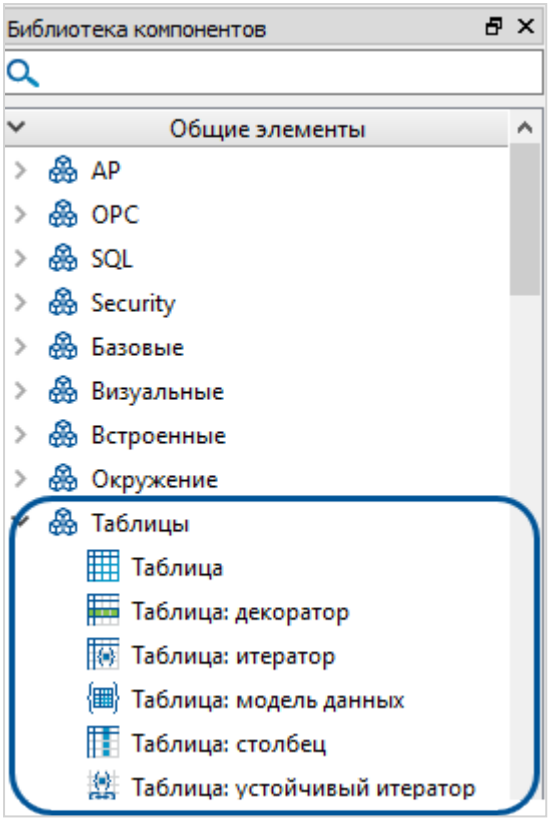
```
cp <имя библиотеки>.so /opt/SePlatform/SePlatform.HMI
```

ОБРАТИТЕ ВНИМАНИЕ

Версия SDK, на которой выполнялась сборка библиотеки, должна быть такой же, какая используется в SePlatform.HMI.

После подключения библиотеки в библиотеку компонентов SePlatform.HMI добавится юнит **Таблицы** с компонентами:

- **Таблица: модель данных**
- **Таблица**
- **Таблица: столбец**
- **Таблица: декоратор**
- **Таблица: итератор**
- **Таблица: устойчивый итератор**



Чтобы сформировать таблицу данных, предварительно выберите способ заполнения таблицы. Таблица может заполняться данными, полученными от источника, либо данными, введенными вручную с помощью скриптов SePlatform.Om или JavaScript.

В любом из выбранных способов используется компонент **Таблица: модель данных**. Модель данных хранит в себе полученные данные в табличном виде.

Чтобы графически представить на экранной форме данные, хранящиеся в модели, используйте компонент **Таблица**.

Заполнение таблицы данными от источника

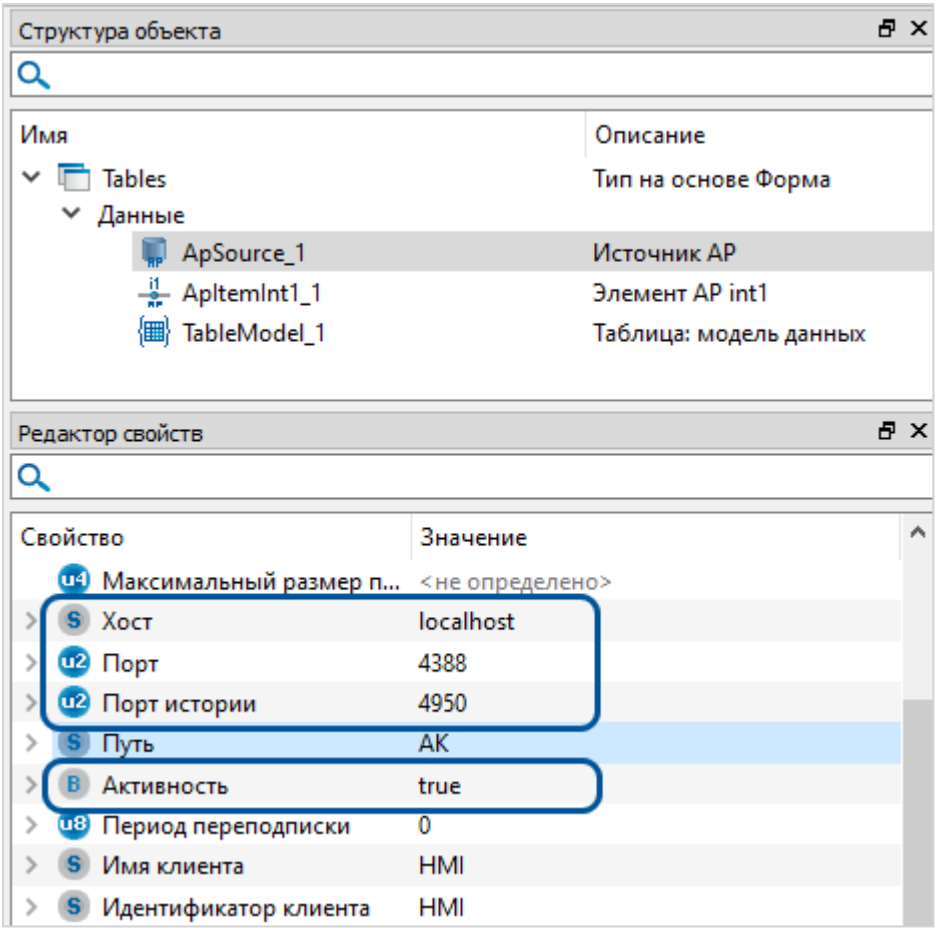
В качестве источника, к примеру, используйте SePlatform.Data Server, с которого можно запросить оперативные и исторические события по TCP. Каждое событие, сгенерированное на источнике, будет помещаться в таблицу модели данных.

ОБРАТИТЕ ВНИМАНИЕ

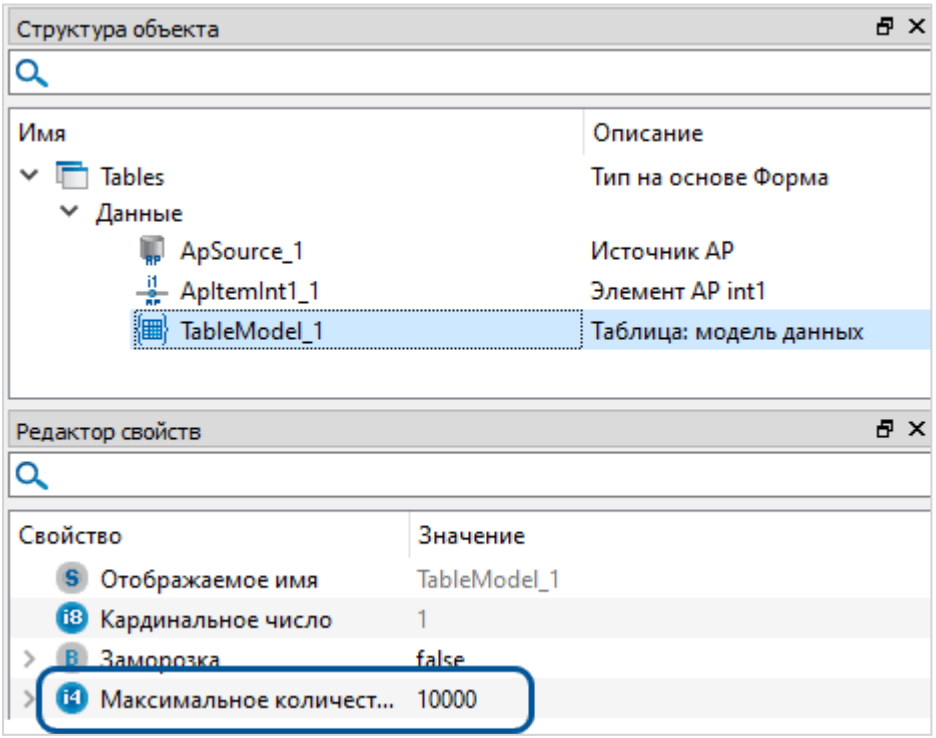
Вы не можете изменить количество строк и столбцов в таблице модели данных, так как это выполняет компонент **Запрос алармов** на основе данных, полученных от источника.

Чтобы настроить заполнение таблицы модели данных событиями, полученными с источника:

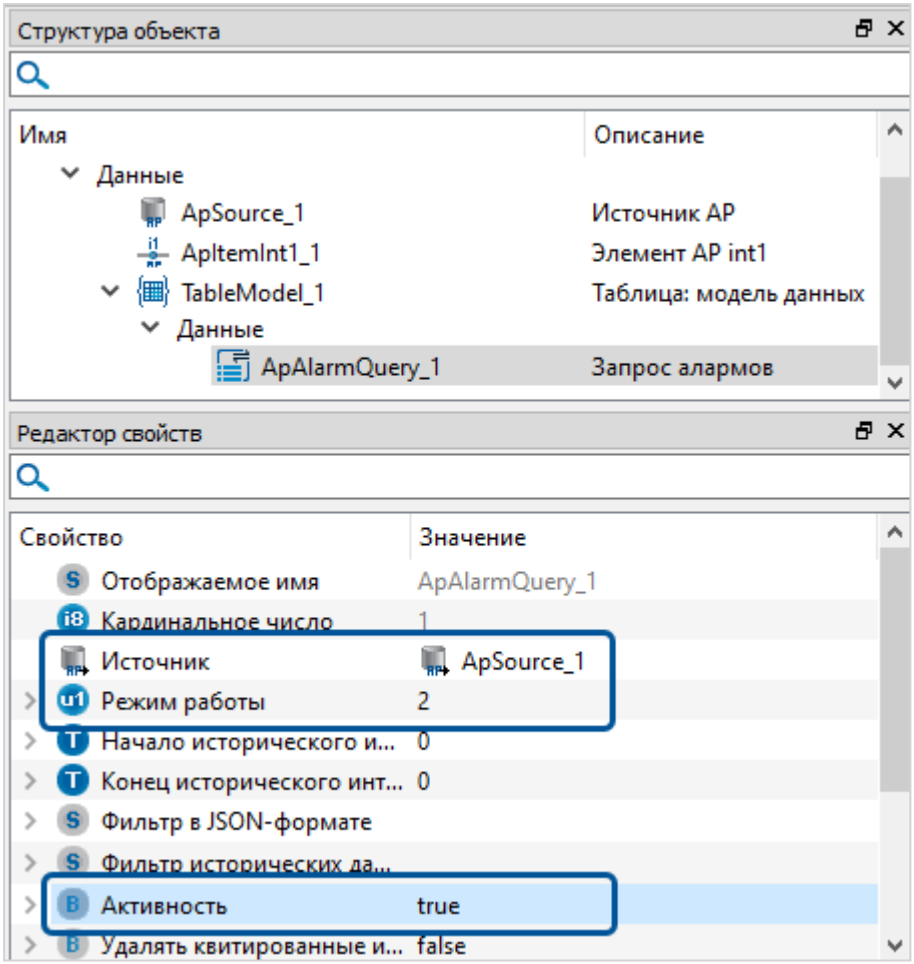
1. Добавьте компонент **Источник AP**. Задайте значения в свойствах Хост, Порт, Порт истории (если планируете запрашивать исторические события), Активность.



2. Добавьте компонент **Таблица: модель данных**. Укажите максимальное число строк таблицы в свойстве Максимальное количество рядов данных.



3. Добавьте компонент **Запрос алармов** дочерним компоненту **Таблица: модель данных**. Укажите источник данных в свойстве Источник, задайте режим работы компонента в зависимости от вида запрашиваемых событий (**«1»** - исторические события, **«2»** - оперативные события) и активируйте компонент в свойстве Активность.



4. Пропишите скрипт, который будет выполнять запрос данных от источника (функция Reload() компонента **Запрос алармов**) и чтение полученных данных моделью данных (функция BeginReadAsync() компонента **Таблица: модель данных**).
5. Добавьте компонент **Таблица** для визуализации данных из модели (см. ниже).

Для самостоятельного воспроизведения стандартных функций SePlatform.HMI.Alarms (подгрузка исторических данных, удаление квитированных событий и т.д.) используйте свойства и функции компонента **Запрос алармов**.

Ручное заполнение таблицы

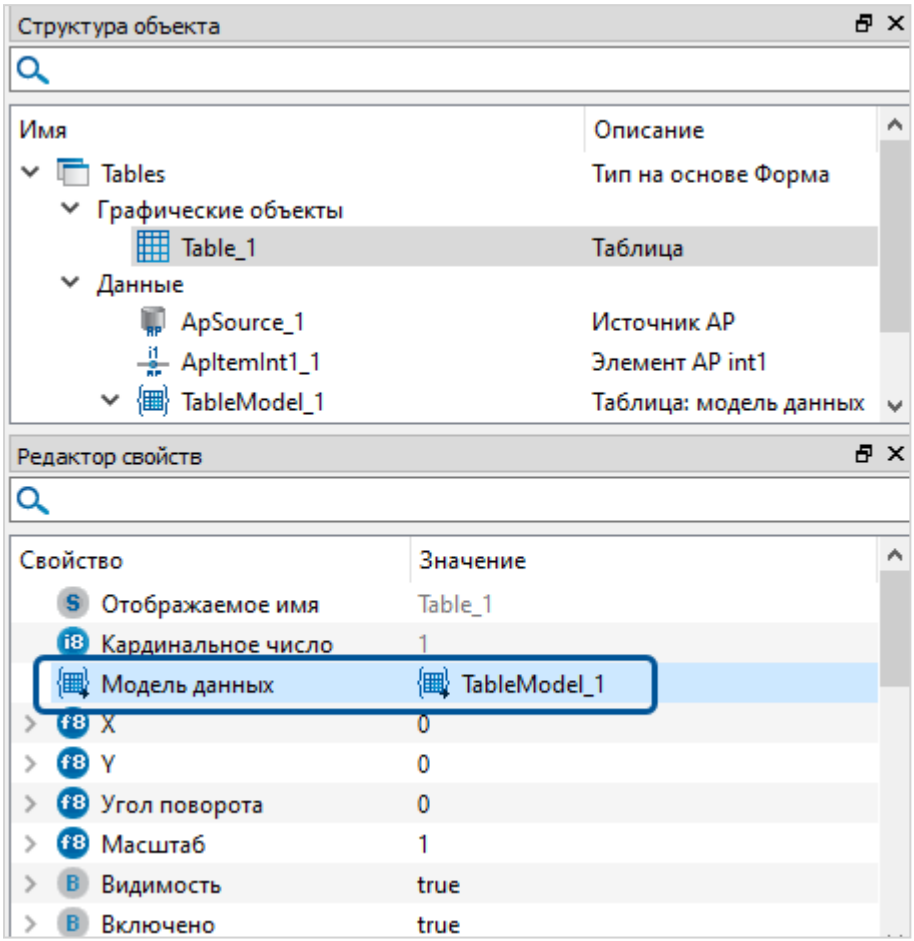
Чтобы наполнять таблицу модели данных самостоятельно, используйте функции компонента **Таблица: модель данных**.

Что сформировать таблицу в модели данных, выполните:

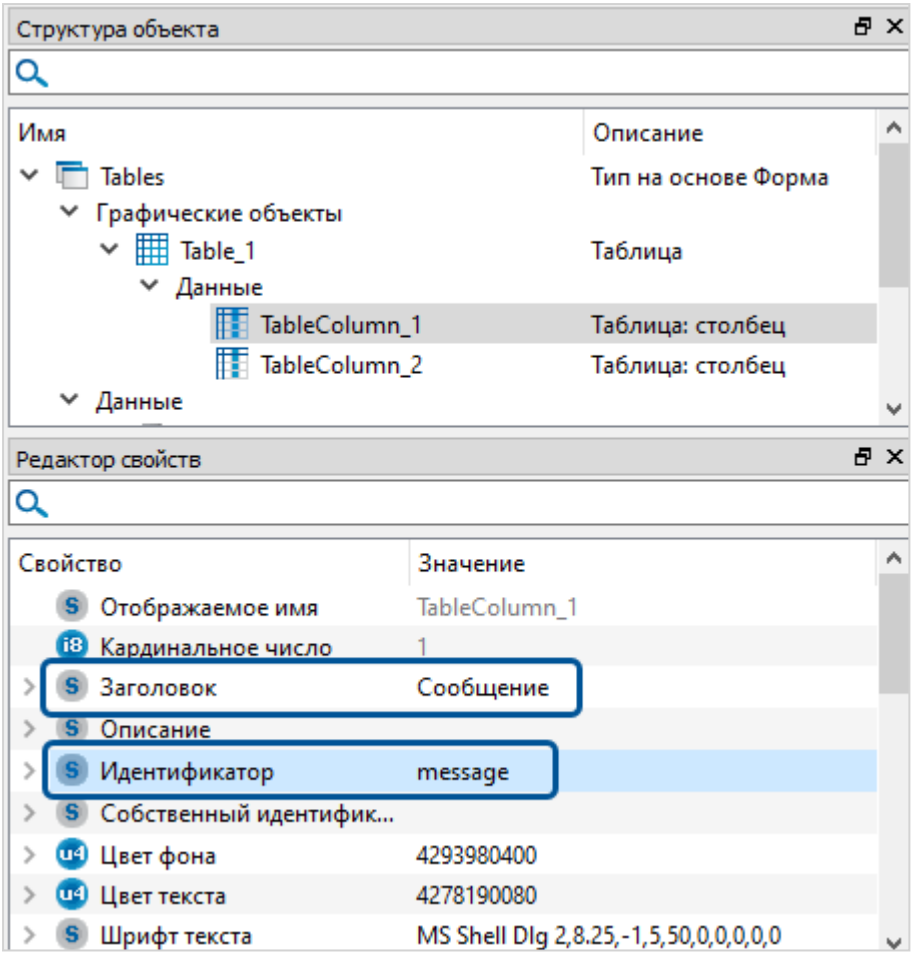
1. Добавьте нужно количество столбцов с помощью функции AddColumn.
2. Добавьте нужно количество строк с помощью функции AddRow.
3. Заполните ячейки таблицы данными с помощью функции SetCellData.
4. Добавьте компонент **Таблица** для визуализации данных из модели (см. ниже).

Графическое представление данных из таблицы модели данных

Чтобы визуализировать данные из таблицы модели добавьте на экранную форму компонент **Таблица**. В свойстве **Модель данных** свяжите таблицу с моделью данных, значения которой будут визуализироваться.



Вы можете настроить отображение в таблице на форме всех данных модели либо только отдельных столбцов. Добавьте компоненту **Таблица** дочерние компоненты **Таблица: столбец**. Для каждого столбца графической таблицы (свойство **Идентификатор**) запишите идентификатор столбца таблицы в модели данных. Идентификатор позволяет указать столбец, из которого в графическую таблицу будут подгружаться данные.



Идентификаторы столбцов таблицы модели соответствуют идентификаторам столбцов в источнике либо указываются вручную при конфигурировании таблицы с помощью скриптов.
Возможные значения идентификатора при получении данных от источника:

Значение	Описание
«source»	Источник события
«time»	Время генерации уведомления о событии
«message»	Сообщение
«severity»	Уровень важности события
«condition_name»	Имя условия генерации события
«subcondition_name»	Имя подусловия генерации события
«quality»	Текущее качество сигнала, изменение которого привело к генерации события
«active_time»	Время перехода состояния события в активное
«actor_id»	Имя пользователя, выполнившего квитирование сообщения о событии

Значение	Описание
«ack»	Признак квитирования события
«active»	Признак активности подусловия, по которому было сгенерировано событие
«cookie»	Специальный идентификатор события, который имеет служебное назначение. Необходим для сведения квитанций, деактиваций с событиями
«ack_time»	Время квитирования
«ack_ required»	Требование квитирования
«value»	Значение сигнала, изменение которого привело к генерации события
«sound»	Звуковой файл, исполняемый при выполнении подусловия генерации сообщения о событии
«area_path»	Относительный тег объекта
«object_id»	Идентификатор объекта на сервере, по которому было сгенерировано событие
«deactive_ time»	Время, когда событие перешло из активного состояния в неактивное

11. Работа с графиками

Чтобы получить возможность представлять данные, принятые от источника, в виде графиков, следует подключить внешнюю библиотеку `seplatform.hmi.charts` (для ОС Windows) и `libseplatform.hmi.charts` (для ОС Linux).
Чтобы подключить библиотеки:

ОС Windows

Разместите файл библиотеки в папке установки SePlatform.HMI.

ОС Linux

Скопируйте библиотеку в папку SePlatform.HMI командой

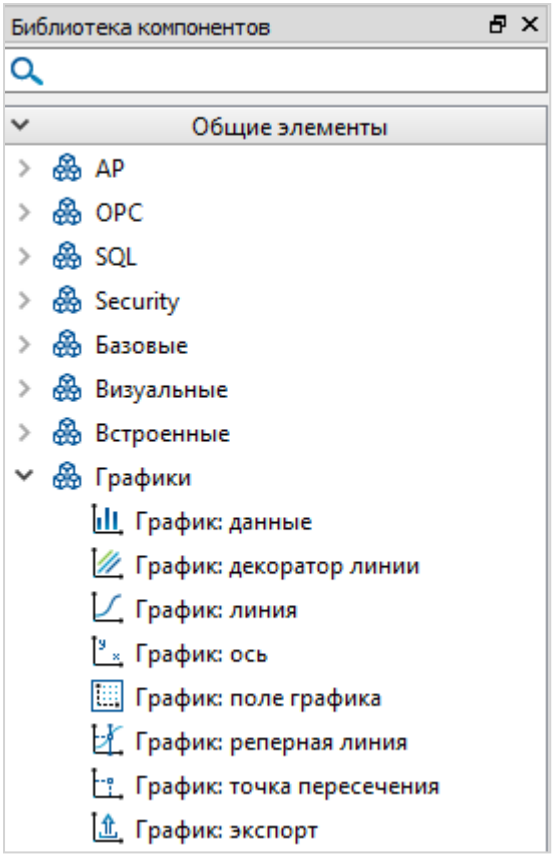
```
cp <имя библиотеки>.so /opt/SePlatform/SePlatform.HMI
```

ОБРАТИТЕ ВНИМАНИЕ

Версия SDK, на которой выполнялась сборка библиотеки, должна быть такой же, какая используется в SePlatform.HMI.

После подключения библиотеки в библиотеку компонентов SePlatform.HMI добавится юнит **Графики** с компонентами:

1. **График: поле графика**
2. **График: ось**
3. **График: данные**
4. **График: реперная линия**
5. **График: линия**
6. **График: декоратор линии**
7. **График: точка пересечения**
8. **График: экспорт**

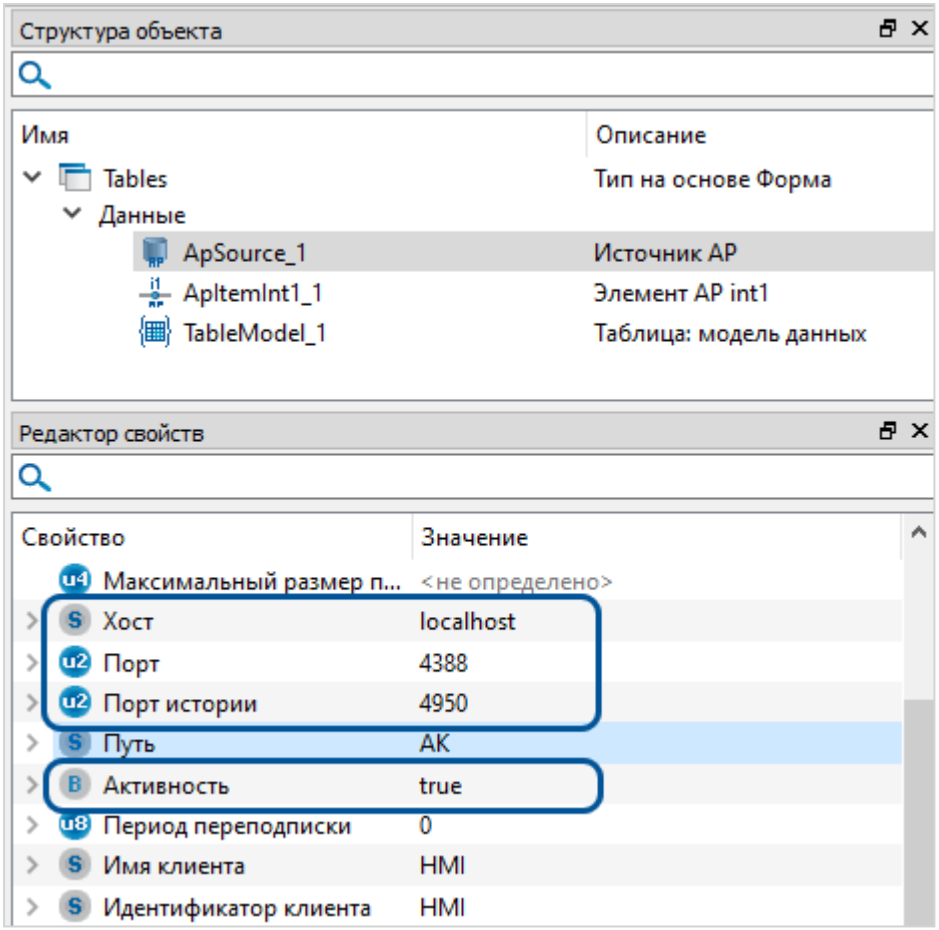


Настройка получения данных от источника

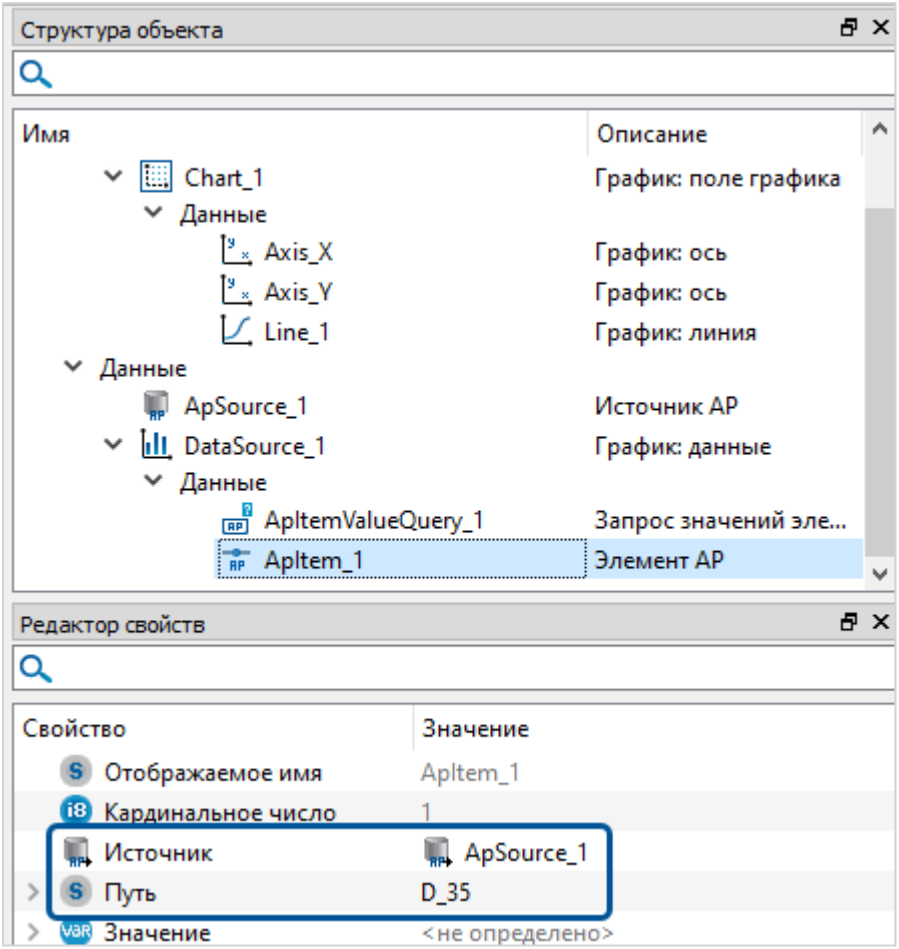
В качестве источника, к примеру, используйте SePlatform.Data Server, с которого можно запросить оперативные и исторические значения по TCP.

Чтобы настроить получение данных от источника, выполните:

1. Добавьте компонент **Источник AP**. Задайте значения в свойствах Хост, Порт, Порт истории (если планируете запрашивать исторические значения), Активность.

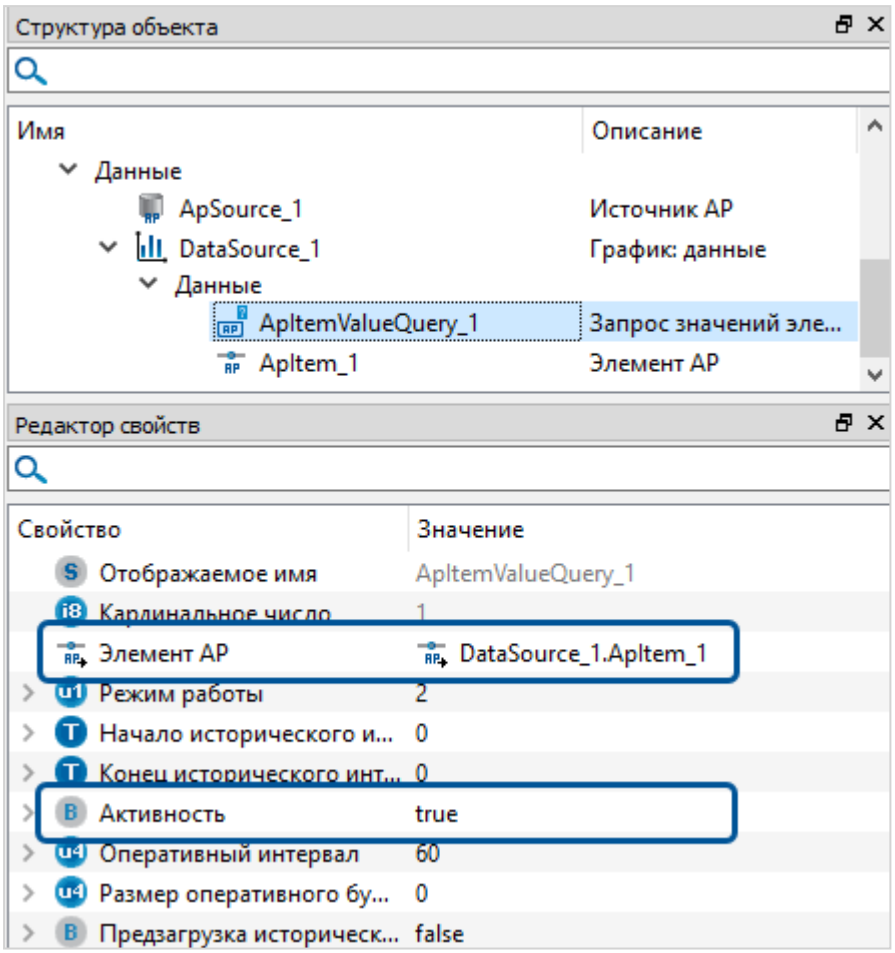

Структура объекта
Имя
Описание
Tables
Тип на основе Форма
Данные
ApSource_1
Источник AP
ApItemInt1_1
Элемент AP int1
TableModel_1
Таблица: модель данных
Редактор свойств
Свойство
Значение
Максимальный размер п...
<не определено>
Хост
localhost
Порт
4388
Порт истории
4950
Путь
AK
Активность
true
Период переподписки
0
Имя клиента
HMI
Идентификатор клиента
HMI

2. Добавьте компонент **Элемент АР**. Укажите источник данных в свойстве Источник и привяжите компонент к сигналу в свойстве Путь.



3. Добавьте компонент **График: данные**. С помощью компонента можно оперировать значениями только одного сигнала. Вы можете использовать компонент в качестве контейнера для объединения компонентов, работающих с одной линией.

4. Добавьте компонент **Запрос значений элемента АР** дочерним компоненту **График: данные**. Укажите элемент (сигнал), с которого компонент будет принимать данные, в свойстве **Элемент АР** и активируйте компонент в свойстве **Активность**. Задайте режим работы компонента в зависимости от вида запрашиваемых значений (**«1»** - исторические значения, **«2»** - оперативные значения).

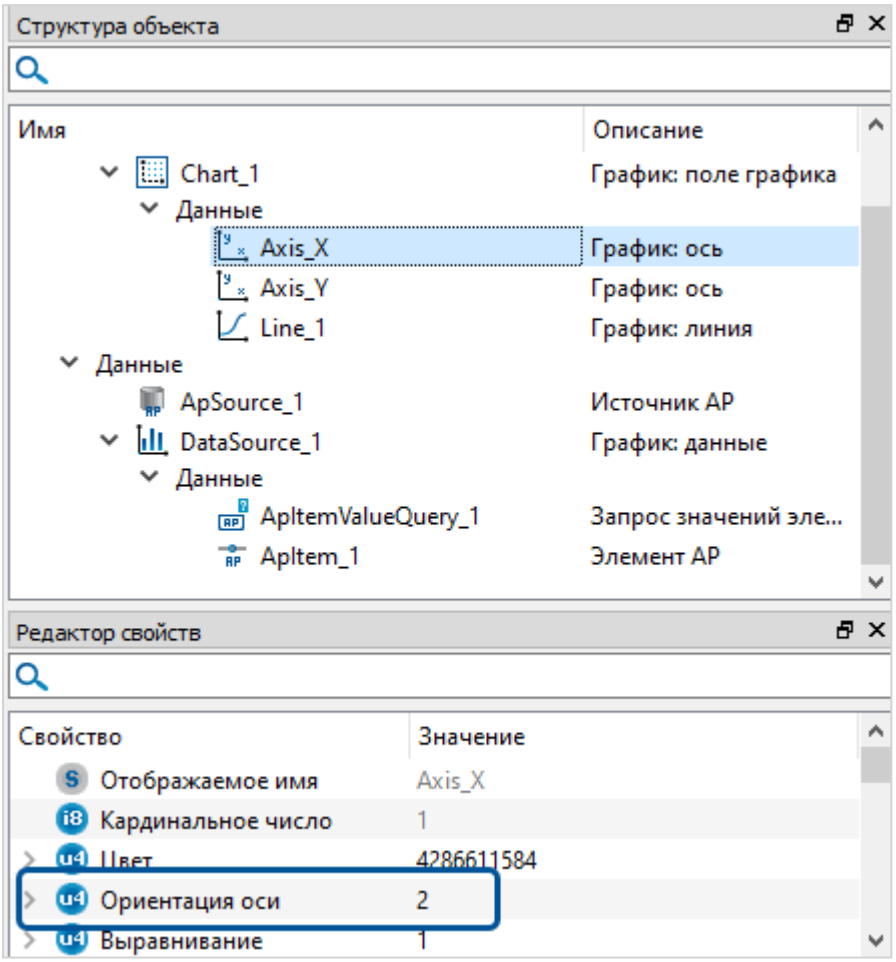


Отображение одного графика на экранной форме

Чтобы вывести значения, полученные от источника, в трендовом виде, выполните:

1. Добавьте компонент **График: поле графика**. Настройте его внешний вид с помощью свойств.

2. Добавьте компоненты **График: ось** дочерними компоненту **График: поле** по количеству нужных осей (минимальное количество - 2). Задайте значения свойства Ориентация оси для каждой оси.



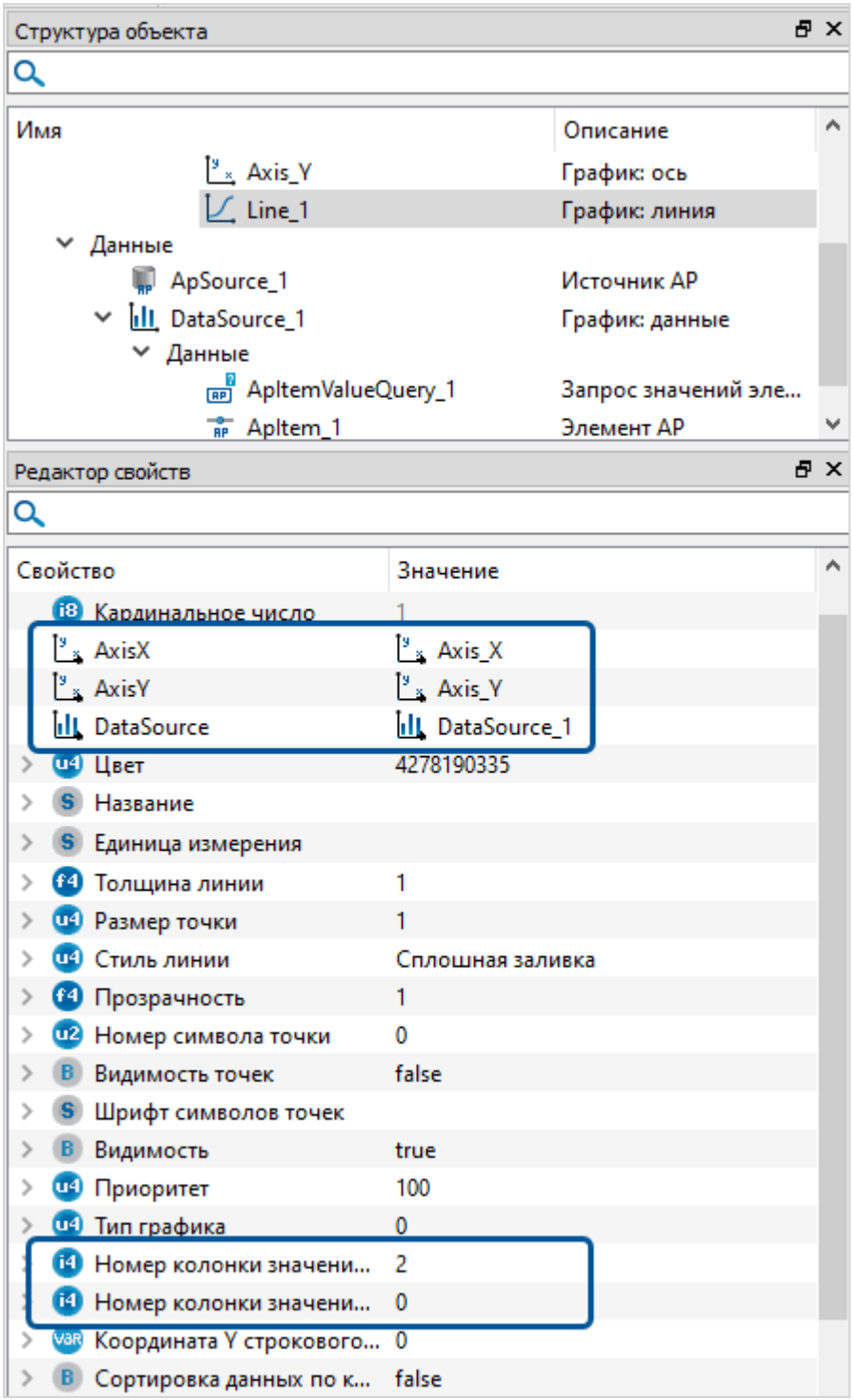
3. Добавьте компонент **График: линия**. Поскольку компонент отвечает за вывод графика только по одному сигналу, для удобства можно расположить компонент дочерним для компонента **График: данные**.

Для линии:

3.1. Привяжите линию к сигналу с помощью свойства DataSource;

3.2. Укажите какие именно данные сигнала будут выводиться на графике в свойствах Номер колонки значений X и Номер колонки значений Y. Указывайте номера колонок, используемых в источнике: **«0»** - колонка значений сигнала (value), **«2»** - значения времени (timestamp), **«2»** - значения качества (quality);

3.3. Привяжите линию к осям с помощью свойств AxisX и AxisY.



4. Чтобы отобразить график оперативных данных, пропишите скрипт, который будет выполнять запрос оперативных данных от источника (функция Reload() компонента **Запрос значений элемента АР**) и чтение полученных данных копонентом **График: данные** (функция BeginReadAsync() компонента **График: данные**).

   Чтобы отобразить график исторических данных, укажите интервал запроса в свойствах Начало исторического интервала и Конец исторического интервала компонента **Запрос значений элемента АР**.

Отображение нескольких графиков на экранной форме

Вложите подготовленный компонент **График: данные** в компонент **График: поле графика**. Добавьте компоненту **График: поле графика** столько дочерних компонентов **График: данные**, сколько графиков вам необходимо отобразить на трендовом поле.

Для получения данных по одному сигналу и отображения их в виде графика один компонент **График: данные** должен быть связанным или содержать:

1. Один компонент **Запрос значений элемента АР**.
2. Один компонент **График: линия** (может не быть дочерним).
3. Один компонент **Элемент АР** (может не быть дочерним).

12. Интерфейс командной строки

CLI-утилита (Интерфейс командной строки) **seplatform.hmi.cli** входит в состав SePlatform.HMI и предназначена для управления проектами SePlatform.HMI через командную строку. CLI-утилита предоставляет возможность автоматизировать различные задачи, такие как компиляция проекта и выгрузка метаинформации. В данном разделе рассмотрены ключевые аспекты работы с SePlatform.HMI через командную строку.

12.1. Подготовка к работе

Перед использованием команд **seplatform.hmi.cli** убедитесь, что вы находитесь в директории установки SePlatform.HMI. Для этого выполните следующую команду (пример для ОС Windows):

```
cd C:\"Program Files"\SePlatform\SePlatform.HMI
```

Структура командной строки

Общий формат командной строки выглядит следующим образом:

```
seplatform.hmi.cli COMMAND [OPTIONS]
```

- **«seplatform.hmi.cli»**: Название утилиты.
- **«COMMAND»**: Команда для выполнения определенной операции.
- **«OPTIONS»**: Дополнительные параметры и настройки для команды.

12.1.1. Команды

CLI-утилита **seplatform.hmi.cli** предоставляет две основные команды для управления процессом работы с проектами SePlatform.HMI.

- **«compile»**. Команда compile предназначена для компиляции проекта.
- **«extract»**. Команда extract позволяет выгрузить метаинформацию (метаописания из модулей и проектов).

12.1.1.1. Компиляция проекта

Для компиляции проекта используйте команду **«compile»**. Команда **«compile»** запускает процесс компиляции проекта SePlatform.HMI. Во время компиляции все необходимые файлы и ресурсы проекта обрабатываются и преобразуются в исполняемые компоненты, готовые к работе.

Для использования вместе с командой **«compile»** доступны следующие опции:

- **«--solution-path»**: Путь к файлу проекта SePlatform.HMI.
- **«--output-format»**: Опция для указания формата вывода информации о компиляции. Может быть Plain (текстовый формат) или формат JSON.
- **«--export-binom»**: Опция экспорта проекта SePlatform.HMI в формат binom.

ПРИМЕР

Выполнить компиляцию проекта SePlatform.HMI с использованием CLI-утилиты **seplatform.hmi.cli** с выводом информации о компиляции в json формате.

```
seplatform.hmi.cli --solution-path C:\Projects\Project_HMI\Test_HMI.hmi --output-format json compile
```

ПРИМЕР

Выполнить компиляцию и экспорт проекта SePlatform.HMI в формат binom с использованием CLI-утилиты **seplatform.hmi.cli**.

```
seplatform.hmi.cli --solution-path C:\Projects\Project_HMI\Test_HMI.hmi --output-format json compile --export-binom
```

12.1.1.2. Выгрузка метаинформации

Для выгрузки метаинформации (метаописаний из модулей и проектов) используйте команду **«extract»**. Команда **«extract»** предназначена для выгрузки текстового описания встроенных и нативных юнитов в формате JSON или Plain.

Для использования вместе с командой **«extract»** доступны следующие опции:

- **«--language»**: Указывает язык локализации, используемый в ходе работы команды и язык выгружаемых метаописаний. Допустимые значения: **«ru_RU»** (русский), **«en_US»** (английский).
- **«--solution-path»**: Путь к проекту SePlatform.HMI. Если не задано, метаописания извлекаются только из загруженных нативных и встроенных модулей. Если задано, то в том числе из проекта SePlatform.HMI.
- **«--output-format»**: Указывает формат отображения вывода. Доступные значения: plain, json. Значение по умолчанию: plain.
- **«--output-file»**: Путь к выходному JSON-файлу метаописаний. Этот параметр является обязательным.
- **«--exclude-unit <UUID>»**: Опция для исключения указанного юнита с помощью его уникального идентификатора.

ПРИМЕР

Выполнить выгрузку метаинформации проекта SePlatform.HMI с использованием CLI-утилиты **seplatform.hmi.cli** с выводом информации в json формате.

```
seplatform.hmi.cli extract --solution-path C:\Projects\Project_HMI\Test_HMI.hmi --output-format json --output-file  C:\Projects\Project_HMI\Test.json
```

12.1.2. Форматы вывода

CLI-утилита **seplatform.hmi.cli** предоставляет возможность выбора формата вывода информации: формат Plain и формат JSON.

12.1.2.1. Формат Plain

Опция «**--output-format plain**» активирует простой текстовый формат. В этом формате сообщения выводятся следующим образом:

```
[SOURCE] MESSAGE-TYPE: UNIT, ELEMENT, TEXT
```

Где:

- «**SOURCE** »- источник сообщения, например, сама команда (compile или extract).
- «**MESSAGE-TYPE**» - тип сообщения: error, warning, info или hint.
- «**UNIT** »- наименование юнита.
- «**ELEMENT** »- полный путь (относительно юнита) к элементу, ассоциированному с сообщением.
- «**TEXT** »- текст сообщения.

ПРИМЕР

Выполнить компиляцию проекта SePlatform.HMI с использованием CLI-утилиты **seplatform.hmi.cli** с выводом информации о компиляции в plain формате.

Введем в командной строке следующую команду:

```
seplatform.hmi.cli --solution-path C:\Projects\Project_HMI\Test_HMI.hmi --output-
format plain compile
```

Результат вывода в plain формате:

```
[compile] info: < Компиляция началась >
[compile] info: < Время компиляции: 0.029 секунд >
[compile] info: < Компиляция завершилась успешно >
```

12.1.2.2. Формат JSON

Опция «**--output-format json**» активирует формат вывода в виде JSON-объектов. Каждое сообщение представлено отдельным объектом со следующей json-структурой:

```
{
"source": "источник сообщения",
"message": {
"type": "тип сообщения",
"unit": "наименование юнита",
"element": "полный путь к элементу",
"text": "текст сообщения"
          }
}
```

ПРИМЕР

Выполнить компиляцию проекта SePlatform.HMI с использованием CLI-утилиты **seplatform.hmi.cli** с выводом информации о компиляции в json формате.

Введем в командной строке следующую команду:

```
seplatform.hmi.cli --solution-path C:\Projects\Project_HMI\Test_HMI.hmi --output-
format json compile
```

Результат вывода в json формате:

```
{"source":"compile","message":{"type":"info","text":"< Компиляция началась >"}}
{"source":"compile","message":{"type":"info","text":"< Время компиляции: 0.019 секунд
>"}}
{"source":"compile","message":{"type":"info","text":"< Компиляция завершилась успешно
>"}}
```

12.1.3. Возврат кодов

При взаимодействии с CLI-утилитой **seplatform.hmi.cli** возможны различные коды возврата, которые указывают на результат выполнения операции. Основные коды возврата и их значения:

- **«0»**: Операция выполнена успешно.
- **«1 и выше»**: Операция завершилась с ошибкой.

После выполнения команды **«compile»** или **«extract»**, вы можете использовать **«%errorlevel%»**, чтобы узнать, как завершилась операция.

ПРИМЕР

Выполнить компиляцию проекта и получить код завершения операции:

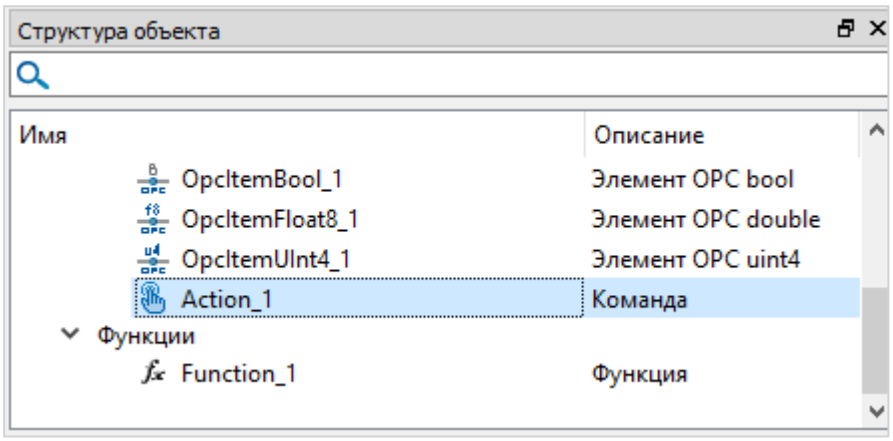
```
REM Компилируем проект
seplatform.hmi.cli compile --solution-path C:\Projects\Project_HMI\Test_HMI.hmi --
output-format plain
REM Проверяем значение %errorlevel%
echo Exit code = %errorlevel%
```

13. Работа с пользовательскими командами

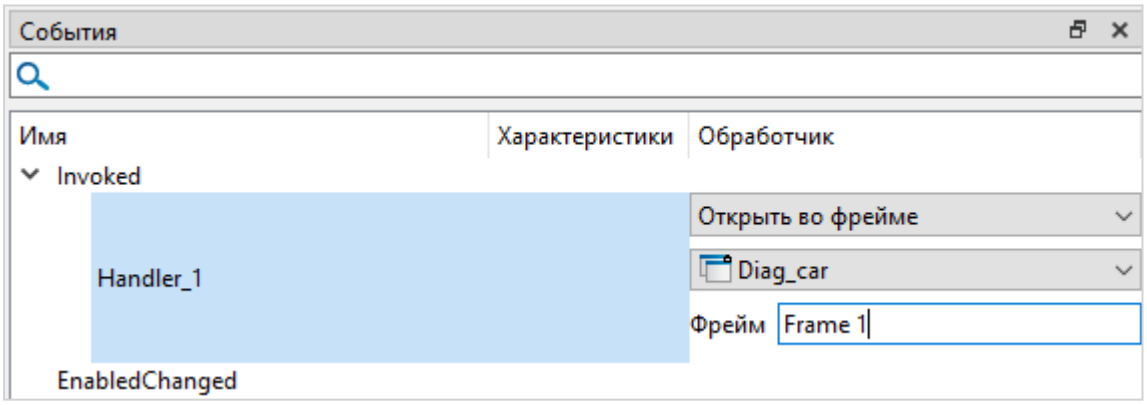
Команда - независимая процедура, определяемая пользователем. Может многократно вызываться из любой части проекта.

13.1. Как создать и использовать пользовательскую команду

1. Чтобы создать новую команду, добавьте компонент **Команда** на экранную форму.



2. Чтобы определить логику выполнения команды, настройте обработчик события Invoked. Это событие срабатывает при активации команды, позволяя выполнить заданную процедуру.



13.1.1. Методы активации команды

Команда может быть активирована одним из следующих способов:

- Прямой вызов через функцию Invoke().

```
Action_1.Invoke();
```

- Активация через свойство Триггер.

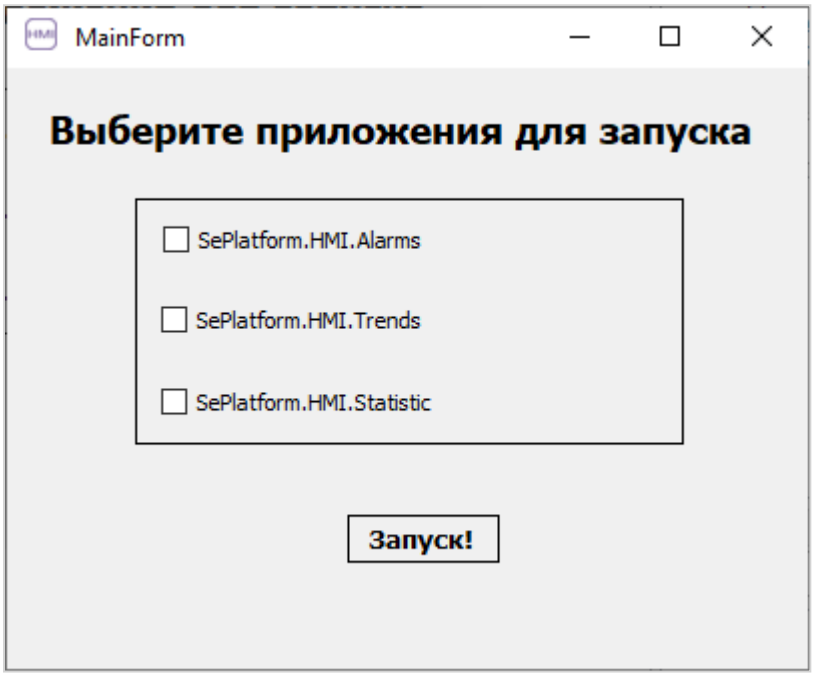
```
Action_1.InvokeTrigger = true;
```

ОБРАТИТЕ ВНИМАНИЕ

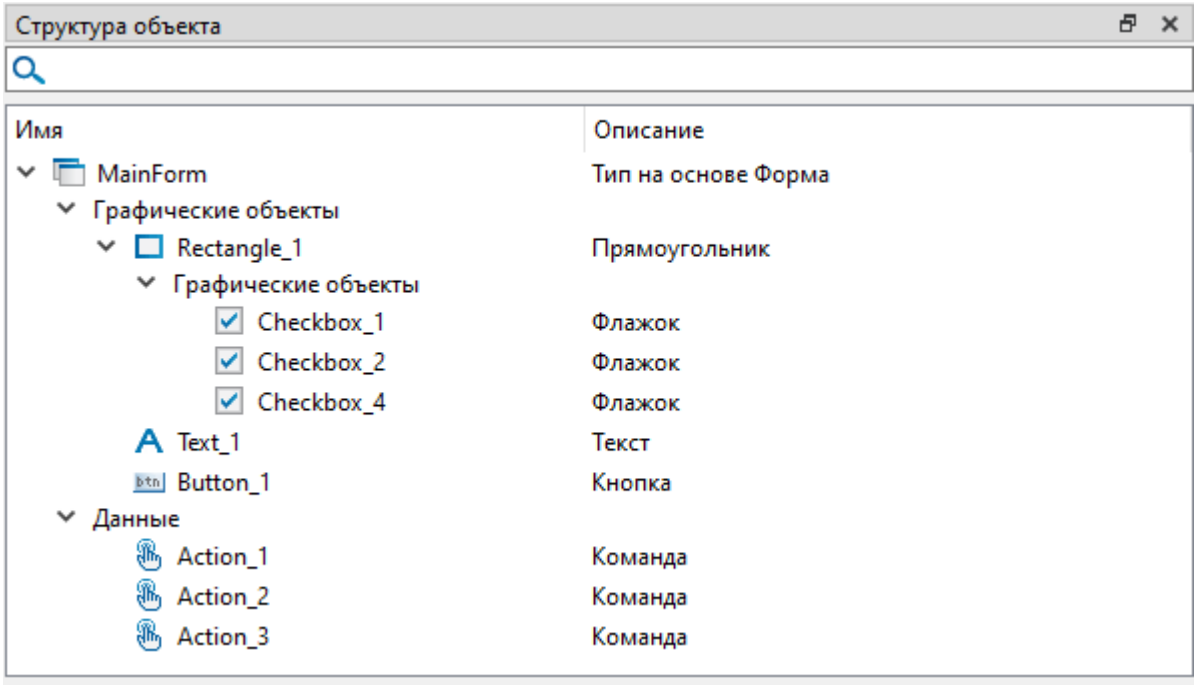
Из-за того, что выполнение команд находится в общей очереди обработчиков событий, то команды вызываемые из скрипта, выполняются отложенно (после выполнения основного кода скрипта). Чтобы вызывать исполнение неких процедур синхронно с кодом скрипта, применяйте функции.

13.2. Демонстрационный пример работы с командами

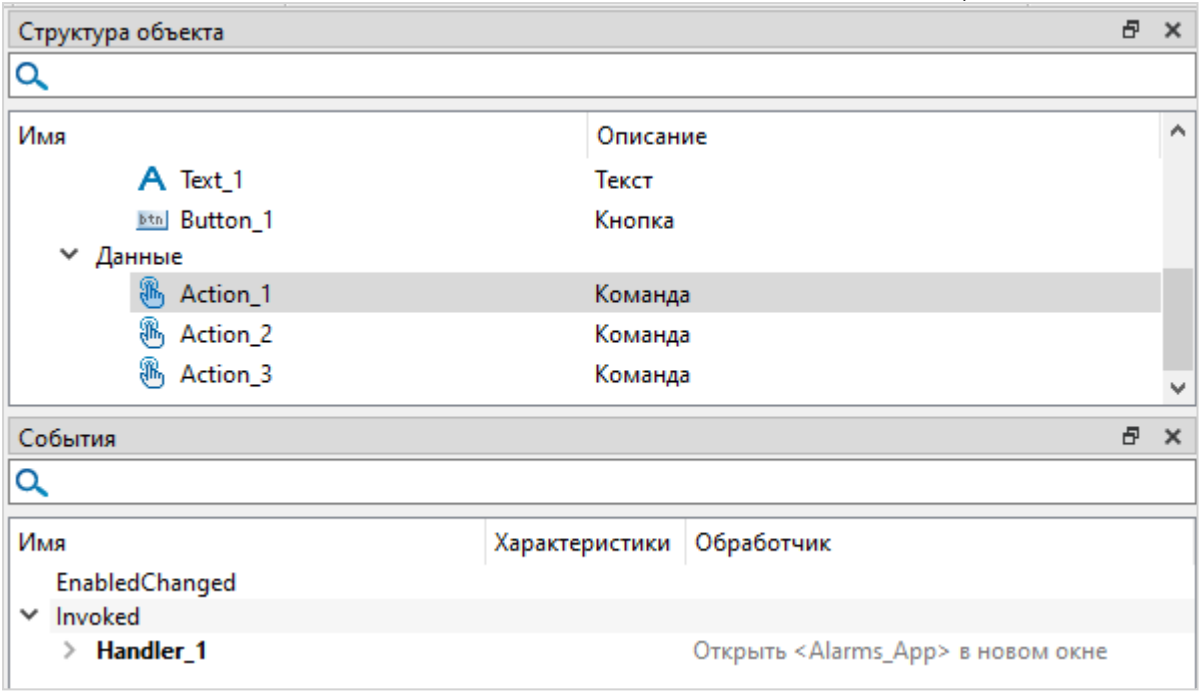
Для демонстрации возможностей компонента **Команда**, ниже рассмотрено приложение (файлы проекта идут в комплекте с документацией), запускающее программы, которые пользователь отметил флагами.



Для создания приложения потребуется 3 **Флажка**, 3 **Команды** и 1 **Кнопка**.



Определите обработчики **Invoked** для каждой команды, который должен запускать программу (SePlatform.HMI.Alarms , SePlatform.HMI.Trends и SePlatform.HMI.Statistics).



Определите код обработчика для кнопки **Запуск**, который будет вызывать нужную команду в зависимости от активности флага (свойство State):

```
if(Checkbox_1.State){
  Action_1.Invoke(); }
if(Checkbox_2.State){
  Action_2.Invoke(); }
if(Checkbox_4.State){
  Action_3.Invoke(); }
```

После нажатия на кнопку должны последовательно запуститься программы, отмеченные флагами.

14. Получение статистических данных

Для получения статистических данных о работе серверов:SePlatform.Data Server, SePlatform.AccessPoint, SePlatform.Historian , SePlatform.Imitator и SePlatform.License Server воспользуйтесь набором компонентов из юнита **ApService**.

Компоненты юнита **ApService**:

- **Источник данных статистики** используется для получения данных статистики от сервера или через срез дерева статистики. Срез дерева статистики - это набор данных о статистике в форматах JSON или XML. Набор данных включает в себя информацию о каждом узле статистики: его имя, тип, значение и дочерние узлы. Подробное руководство по созданию среза дерева статистики доступно здесь.
- **Узел статистики** используется, когда необходимо следить за изменениями значений узлов дерева статистики.
- **Rmap-браузер источника статистики** используется для получения данных об узлах дерева статистики и передачей этих данных графическим компонентам SePlatform.HMI (например, **Дерево**).
- **Json-браузер источника статистики** используется для получения данных об узлах дерева статистики и предоставления их в формате JSON.

Далее более подробно рассматривается описание каждого из компонентов **ApService**, а также принципы их работы.

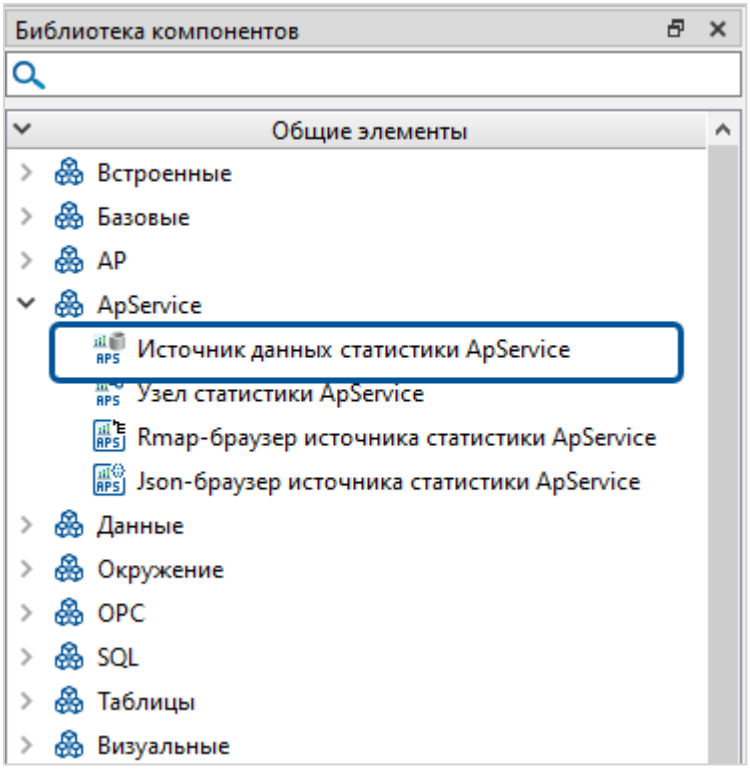
14.1. Источник данных статистики ApService

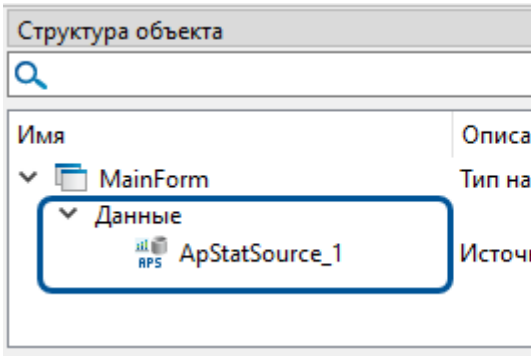
Компонент **Источник данных статистики ApService** представляет собой основной инструмент для получения статистических данных. Вся работа по сбору статистических данных начинается с этого компонента, который затем передает информацию другим компонентам в составе юнита **ApService**.

Источником может быть:

- Сервер.
- Срез дерева статистики.

Компонент **Источник данных статистики ApService** расположен в юните **ApService Библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.





Свойства, методы и события компонента **Источник данных статистики ApService** рассмотрены в справочном руководстве.

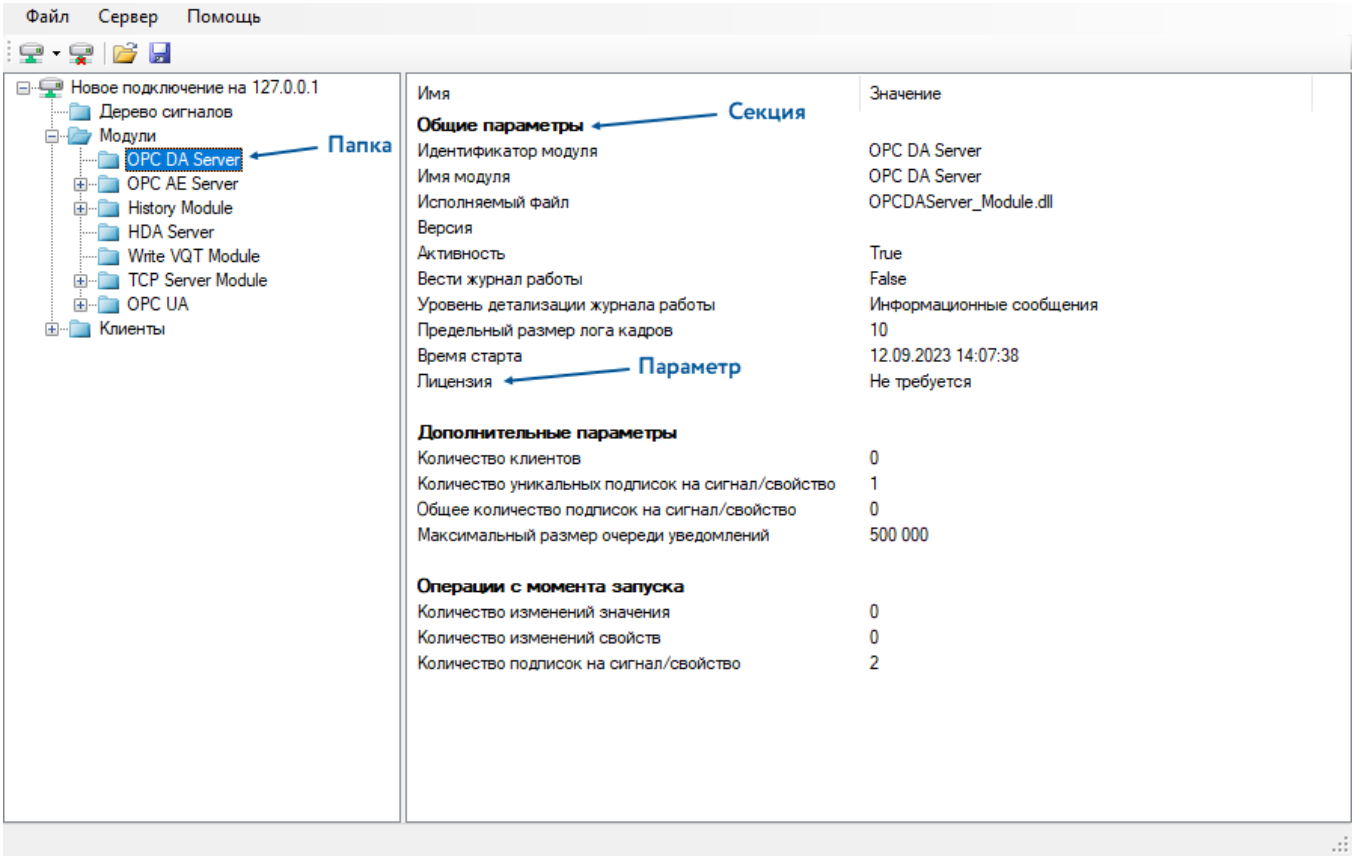
14.2. Узел статистики ApService

Компонент **Узел статистики ApService** позволяет подписываться на обновления значений узлов дерева статистики и следить за изменениями в значениях узлов статистики.

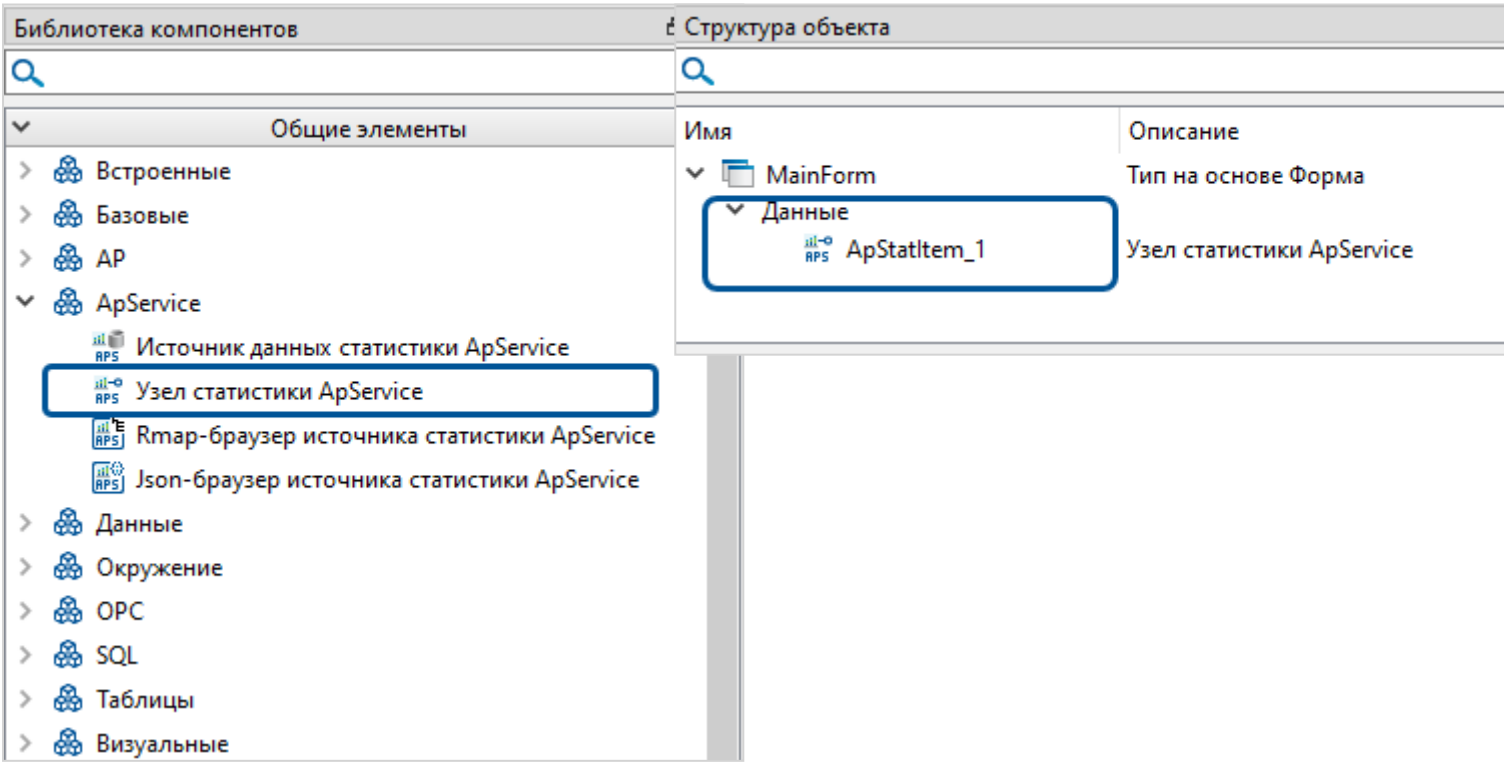
Узлы бывают трех типов:

- Папка - может содержать узлы любых типов и создавать новую вложенность в иерархии дерева статистики.
- Секция - группирует по смыслу несколько других одноуровневых (соседних) узлов, не создавая новой вложенности.
- Параметр- содержит значение параметра статистики и не содержит других узлов.

Ниже представлен пример дерева статистики SePlatform.Data Server, открытого в приложении **Статистика**.



Компонент **Узел статистики ApService** выполняет запросы к компоненту **Источник статистики ApService** и получает значение параметра статистики, тип значения (CDT) и последнее время обновления узла. Как подписаться на изменения значений узла дерева статистики описано здесь.
Компонент **Узел статистики ApService** расположен в юните **ApService Библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



Свойства и события компонента **Узел статистики ApService** рассмотрены в справочном руководстве.

14.2.1. Как подписаться на изменения значений узла дерева статистики

Чтобы компонент **Узел статистики ApService** инициировал запросы к источнику статистики и запрашивал значения параметра статистики:

1. Добавьте объект типа **Узел статистики ApService** на форму.
2. В свойстве Источник статистики ApService укажите источник статистики, с которого вы хотите получать значения параметра статистики.

> ПРИМЕЧАНИЕ
> В проект уже должен быть добавлен объект типа **Источник статистики ApService**.

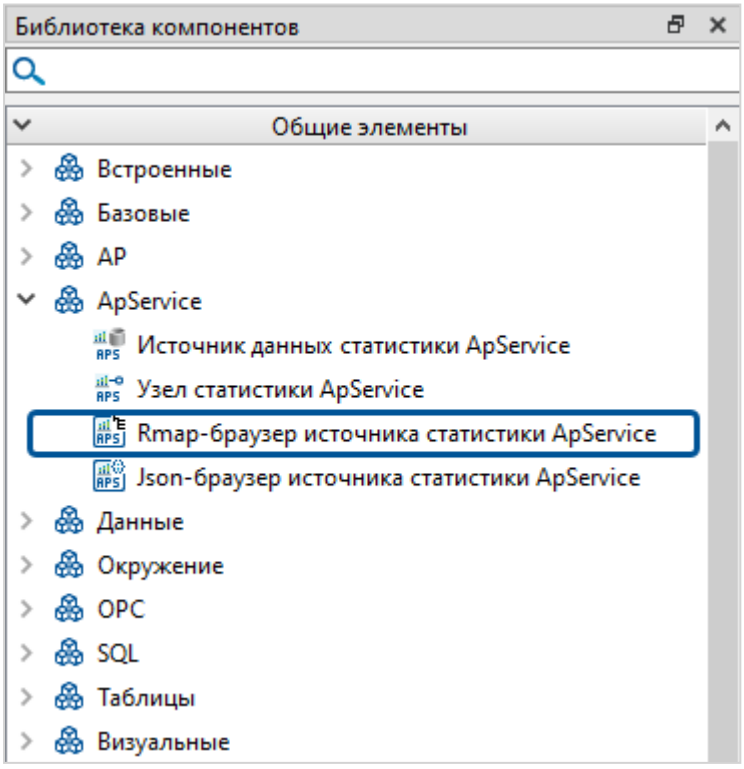
3. Укажите полное имя отслеживаемого узла в свойстве Путь компонента **Узел статистики ApService**.
4. Свойство Активность установите в значение **«true»**.
5. Для получения актуальных значений из узлов используйте событие NodeStateChanged компонента **Узел статистики ApService**.

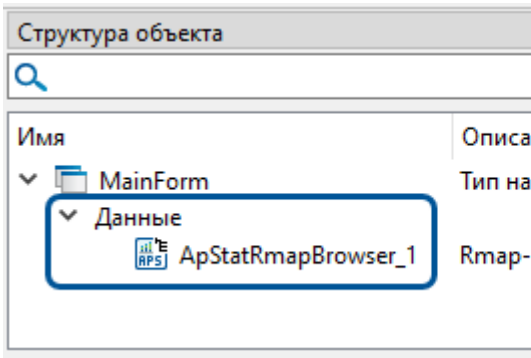
14.3. Rmap-браузер источника статистики ApService

Компонент **Rmap-браузер источника статистики ApService** предназначен для просмотра узлов дерева статистики (далее: браузинг) и передачи результатов браузинга графическим компонентам (например **Дерево**).

Компонент **Rmap-браузер источника статистики ApService** возвращает данные в табличном формате. Для просмотра результатов браузинга подключите **Rmap-браузер источника статистики ApService** к одному из компонентов: **Дерево** или **Таблица**. Как настроить браузинг узлов дерева статистики с помощью компонента **Rmap-браузер источника статистики ApService** описано в демонстрационном проекте.

Компонент **Rmap-браузер источника статистики ApService** расположен в юните **ApService Библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.





Свойства и методы компонента **Rmap-браузер источника статистики ApService** рассмотрены в справочном руководстве.

14.3.1. Какие данные возвращает компонент

Компонент **Rmap-браузер источника статистики ApService** возвращает данные в табличном формате. Используя указанные ниже идентификаторы, вы можете обращаться к этим данным в проекте. Например, вывести их значения в виде древовидной структуры на форме, используя компоненты **Источник данных дерева**, **Дерево** и **Колонка дерева** из юнита **Визуальные**.

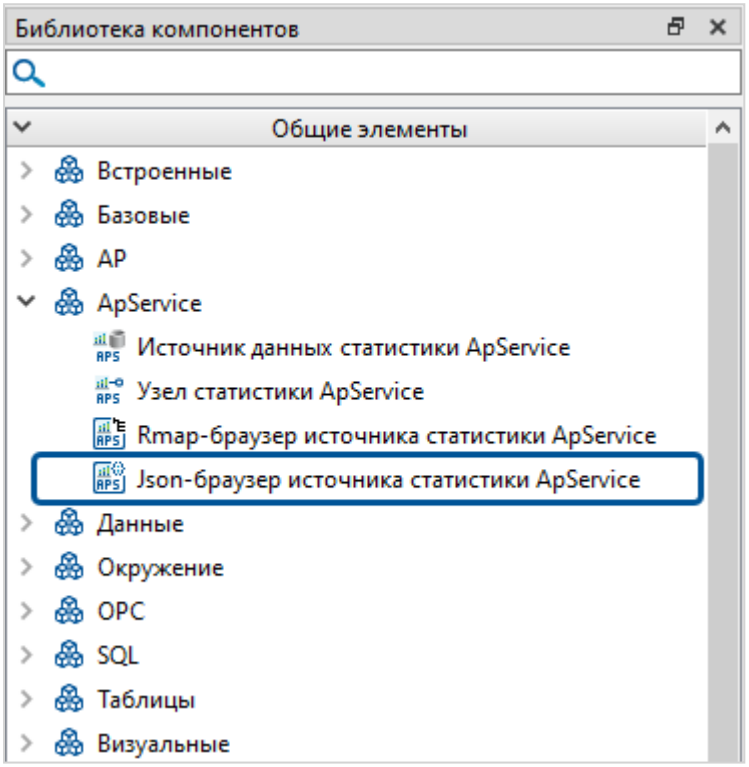
Идентификатор колонки	Тип колонки	Тип возвращаемого значения	Описание
name	1	string	Имя узла.
id	2	string	Идентификатор элемента.
has_children	6	bool	Наличие у узла дочерних узлов.
parent_id	3	string	Идентификатор родительского узла.
node_type	5	uint1	Идентификатор типа узла статистики. › **«0»** - неизвестный тип, › **«1 »**- папка, › **«2»** - секция, › **«3»** - параметр
value_type	5	uint1	Идентификатор типа значения, хранимого в узле статистики. › **«0»** - empty, › **«1»** - bool, › **«2»** - int1, › **«3 »**- uint1, › **«4»** - int2, › **«5 »**- uint2, › **«6»** - int4, › **«7»** - uint4, › **«8»** - int8, › **«9»** - uint8, › **«10»** - float, › **«11»** - double, › **«12»** - timestamp (uint8), › **«13»** - UUID (16 байт), › **«14»** - string

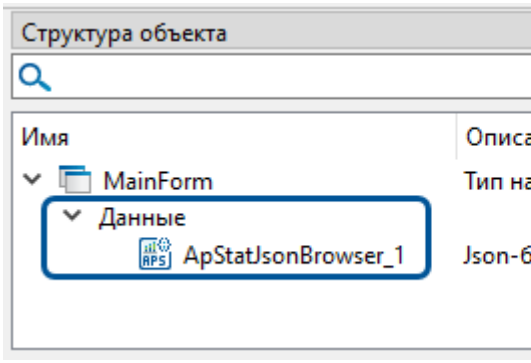
14.4. Json-браузер источника статистики ApService

Компонент **Json-браузер источника статистики ApService** предназначен для просмотра узлов дерева статистики (далее: браузинг) и формирования среза дерева в формате JSON. Подробнее о структуре данных статистики см. здесь.

Как сформировать срез дерева статистики с помощью компонента **Json-браузер источника статистики ApService** описано в демонстрационном проекте.

Компонент **Json-браузер источника статистики ApService** расположен в юните **ApService Библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.





Свойства, методы и события компонента **Json-браузер источника статистики ApService** рассмотрены в справочном руководстве.

14.4.1. JSON-структура данных статистики

Компонент **Json-браузер источника статистики ApService** возвращает данные статистики в виде JSON-массива объектов. Каждый объект включает в себя следующие поля:

Имя поля	Тип данных	Описание	Примечание
Name	string	Сокращенное имя узла статистики.	

Type	number	Идентификатор типа узла.	› **«0»** - неизвестный тип узла, › **« 1»** - узел-папка, › **«2»** - узел-секция, › **«3»** - узел-значение
ValueType	number	Идентификатор типа значения, хранимого в узле.	**«0»** - empty, **«1»** - bool, **«2»** - int1, **«3»** - uint1, **« 4»** - int2, **«5 »**- uint2, **«6»** - int4, **«7»** - uint4, **«8»** - int8, **«9»** - uint8, **«10»** - float, **«11»** - double, **«12»** - timestamp, **«13»** - UUID, **«14»** - string
Value	number, string, boolean или null	Значение, хранимое в узле статистики.	
Children	json-объект	Массив дочерних узлов (рекурсивно представленных в виде аналогичных объектов).	

14.5. Как создать срез дерева статистики

Для создания среза дерева статистики используйте автоматическую генерацию файла статистики с помощью внешних приложений: **Статистика** и SePlatform.HMI.Statistics. Полученную структуру из файла можно скопировать и вставить в свойство **Срез дерева статистики**.

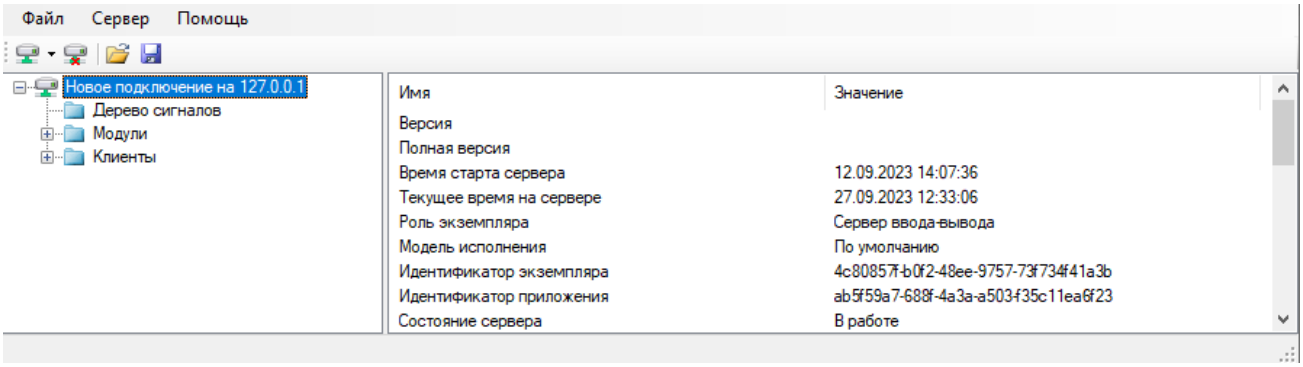
› Для генерации XML структуры используйте сервисное приложение **Статистика**.

› Для генерации JSON структуры используйте прикладное решение SePlatform.HMI.Statistics. Чтобы сформировать срез дерева статистики в формате JSON внутри проекта SePlatform.HMI, воспользуйтесь компонентом **Json-браузер источника данных статистики ApService**. Подробная инструкция доступна в демо-проекте.

14.5.1. Генерация XML структуры

Используйте сервисное приложение **Статистика**.

1. Для сохранения статистики сервера в файл выделите корневой узел сервера в дереве статистики.



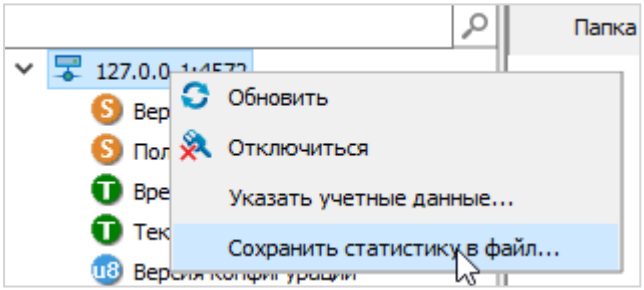
2. Нажмите кнопку или выберите в подменю **Файл** →**Сохранить статистику в файл...**.

Файл будет сохранен с расширением `.stat`, содержащий XML-структуру. Скопируйте ее и вставьте в свойство Срез дерева статистики.

14.5.2. Генерация JSON структуры

Используйте прикладное решение SePlatform.HMI.Statistics.

1. Для сохранения статистики сервера в файл выделите в дереве навигации узел сервера и щелкните по нему правой кнопкой мыши. В открывшемся меню нажмите **Сохранить статистику в файл...**.



2. Файл будет сохранен с расширением `.jstat`, содержащий JSON-структуру. Скопируйте ее и вставьте в свойство Срез дерева статистики.

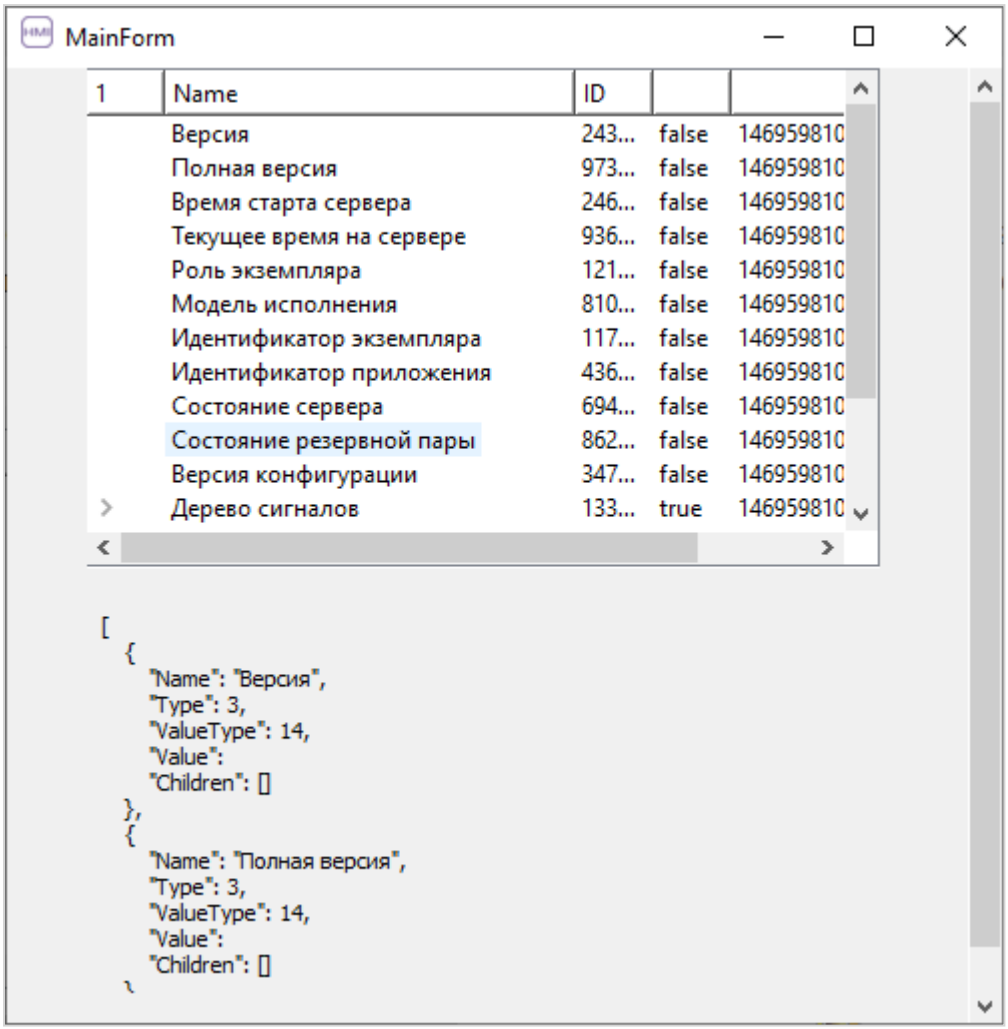
14.6. Как получить статистические данные SePlatform.Data Server (Демо-проект)

Далее кратко описан небольшой пример использования компонентов **Rmap-браузер источника статистики ApService**, **Json-браузер источника данных статистики ApService** и **Источник данных статистики ApService** для получения статистических данных с SePlatform.Data Server в проекте SePlatform.HMI (проект сделан на языке SePlatform.Om).

В примере показано, как:

- Настроить компонент **Источник данных статистики ApService** на получение статистических данных с SePlatform.Data Server.
- Настроить отображение статистических данных SePlatform.Data Server в виде древовидной структуры с использованием графического компонента **Дерево**.
- Настроить компонент **Rmap-браузер источника статистики ApService** для просмотра узлов дерева статистики SePlatform.Data Server.
- Сформировать срез дерева статистики в формате JSON с помощью компонента **Json-браузер источника данных статистики ApService**.

В результате будет подготовлена форма, при открытии которой статистические данные будут получены с SePlatform.Data Server и отображены в двух видах: в виде древовидной структуры и в виде среза дерева статистики в формате JSON.

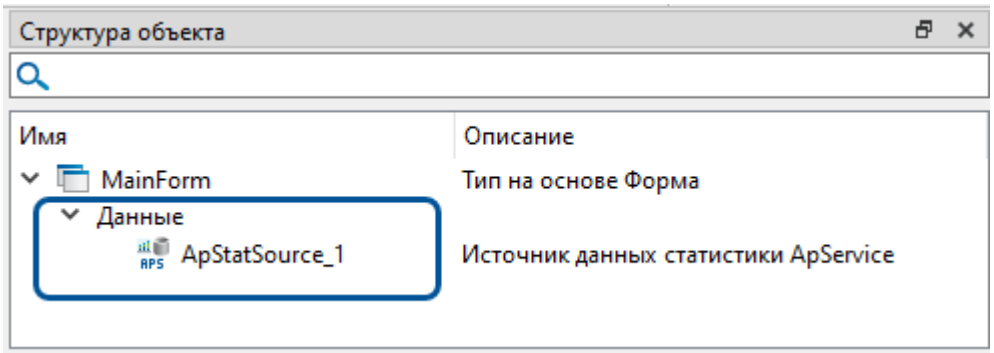


ПРИМЕЧАНИЕ

Перед ознакомлением с проектом-примером предварительно убедитесь, что у вас установлен и настроен сервер для подключения к нему.

14.6.1. Настроить компонент источника данных статистики ApService

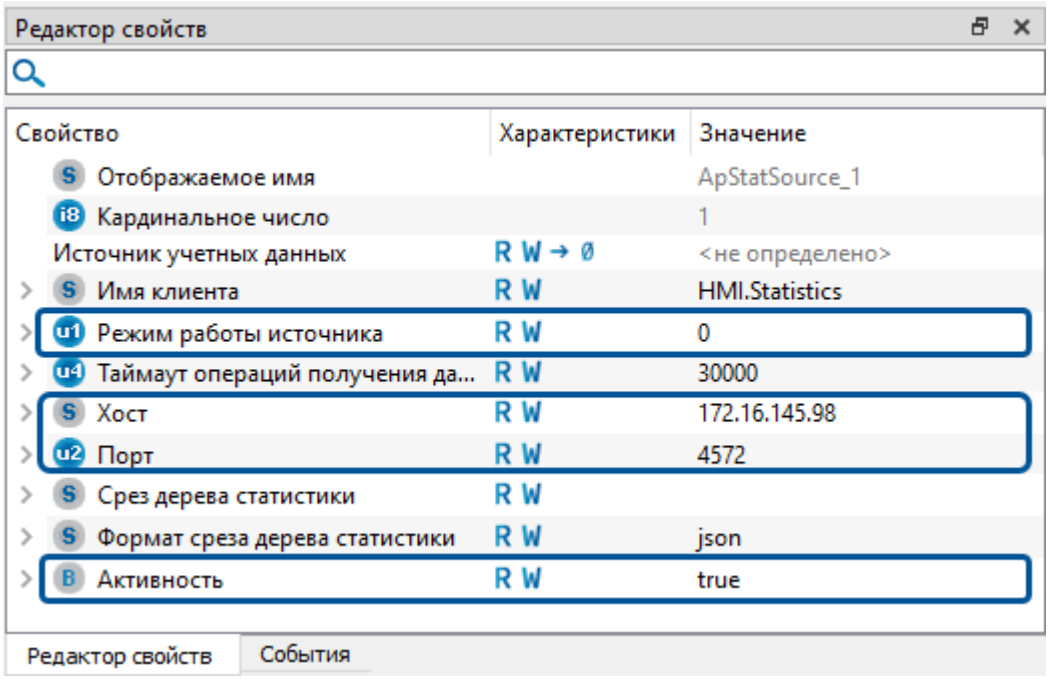
1. Добавьте на форму компонент **`Источник данных статистики ApService`** для взаимодействия с сервером.



2. Укажите адрес сервера и порт в свойствах **Хост** и **Порт**.

   2.1. Активируйте компонент **`Источник данных статистики ApService`**, установив свойство **Активность** в значение **«`true`»**.

   2.2. В свойстве **Режим работы источника** установите режим работы источника на **«`0`»** для приема данных с сервера.

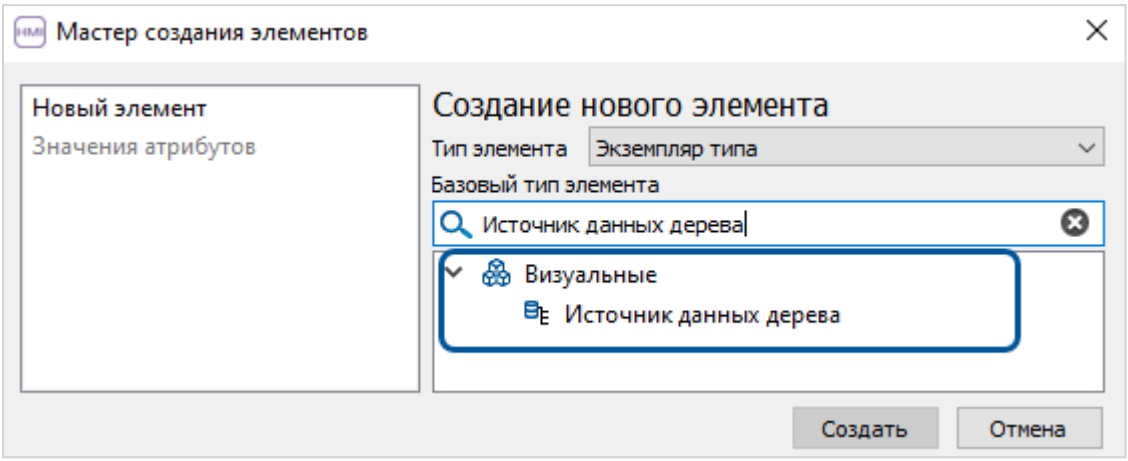


14.6.2. Настроить отображение статистических данных с использованием графического компонента Дерево.

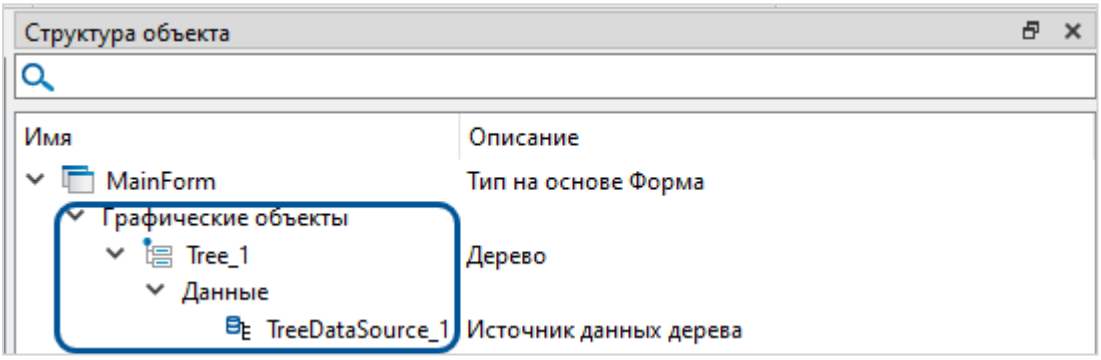
1. Добавьте на экранную форму объект типа **Дерево**.

2. В **Структуре объектов** щелкните правой кнопкой мыши по компоненту **Дерево** → **Создать**. Откроется мастер создания элементов.

2.1. В поле **Тип элемента** оставьте **Экземпляр типа**, **Базовый тип элемента** → **Источник данных дерева**(раздел Визуальные) → **Создать**.



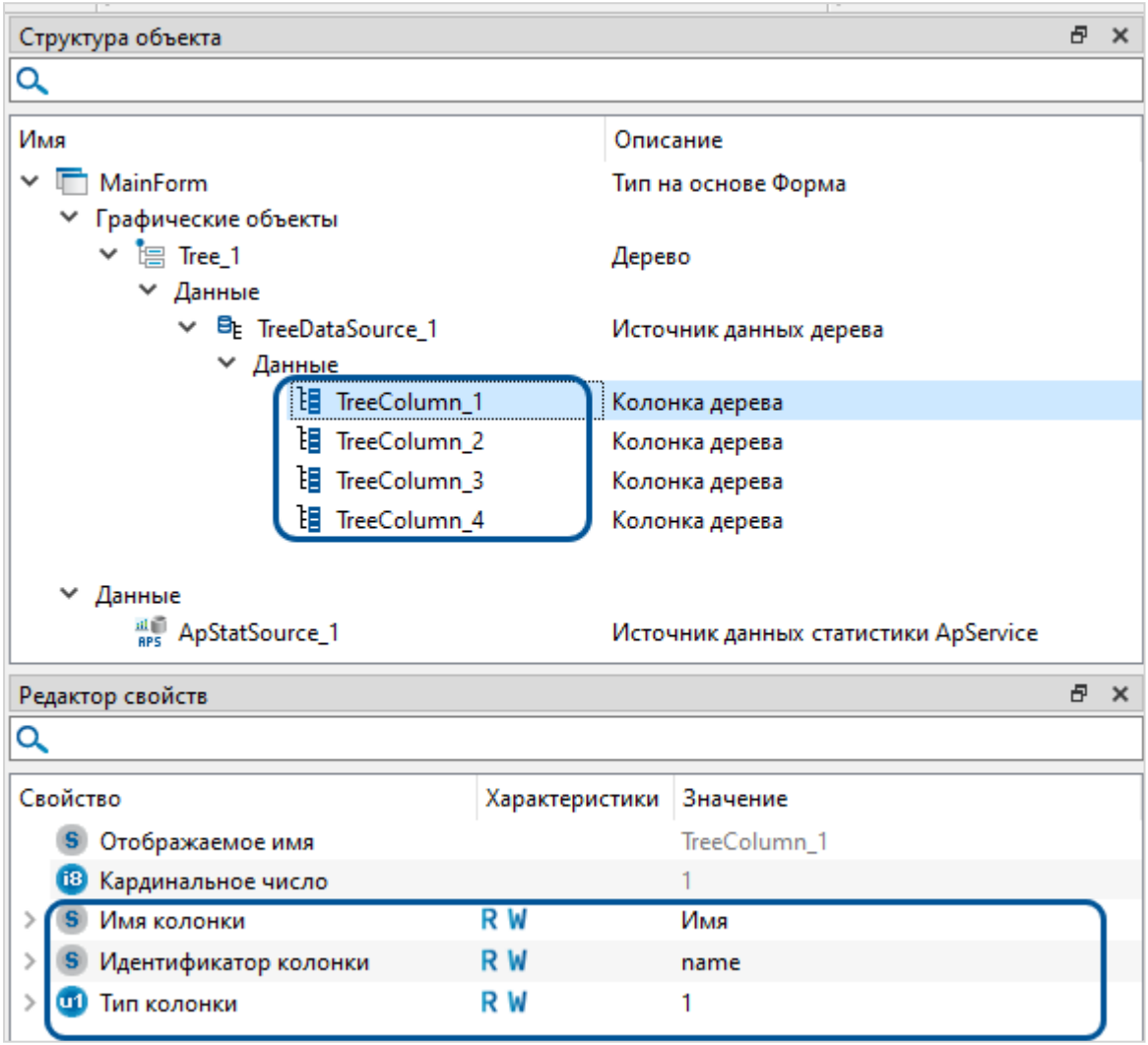
2.2. В **Структуре объектов** добавится компонент **Источник данных дерева** дочерним компоненту **Дерево**.



3. Для объекта **Источник данных дерева** аналогичным образом добавьте **Колонки дерева** с идентификаторами:

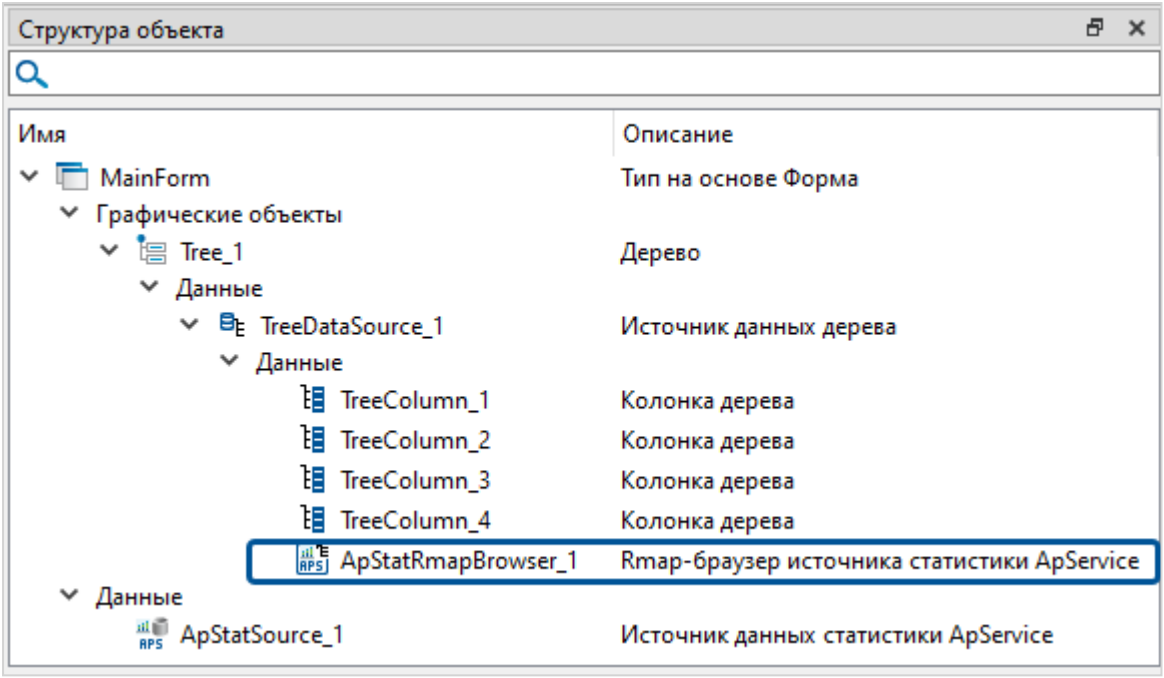
- **«name»**
- **«id»**
- **«parent_id»**
- **«has_children»**

Используйте значения **Идентификатора колонки** и **Тип колонки** в соответствии с таблицей возвращаемых значений в разделе Какие данные возвращает компонент, которая поможет правильно настроить свойства для каждой **Колонки дерева**. **Имя колонки** указывается в произвольном виде.



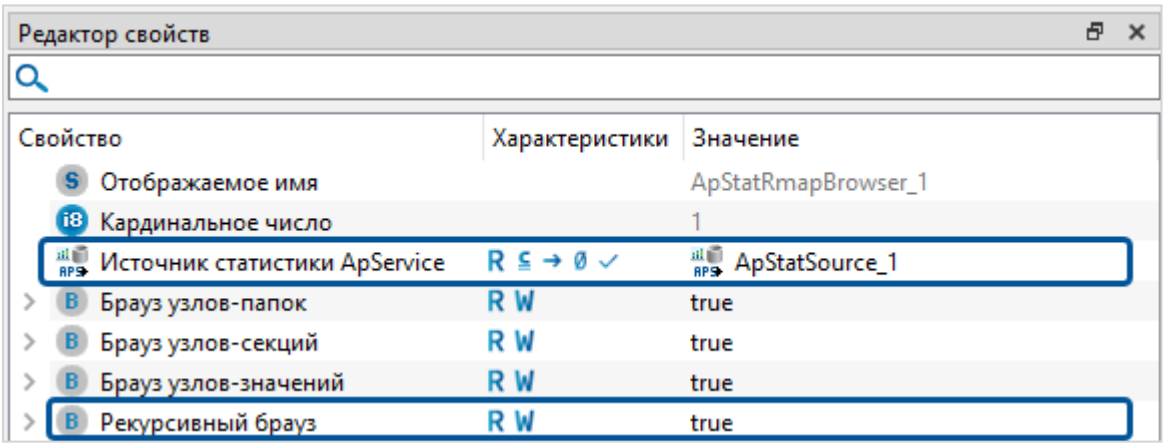
14.6.3. Настроить Rmap-браузер для просмотра узлов дерева статистики SePlatform.Data Server.

1. Для просмотра узлов дерева статистики добавьте на форму компонент **Rmap-браузер источника статистики ApService** дочерним **Источнику данных дерева**.



2. В редакторе свойств у объекта **`Rmap-браузер источника статистики ApService`** укажите ссылку на добавленный ранее компонент **`Источник данных статистики ApService`**. Для этого нажмите правой кнопкой мыши по свойству Источник статистики ApService → **`Сослаться`** → **`ApStatSource_1`**.

В свойстве Рекурсивный брауз установите значение **«`true`»** для просмотра узлов на всех уровнях иерархии дерева статистики, включая все подуровни.

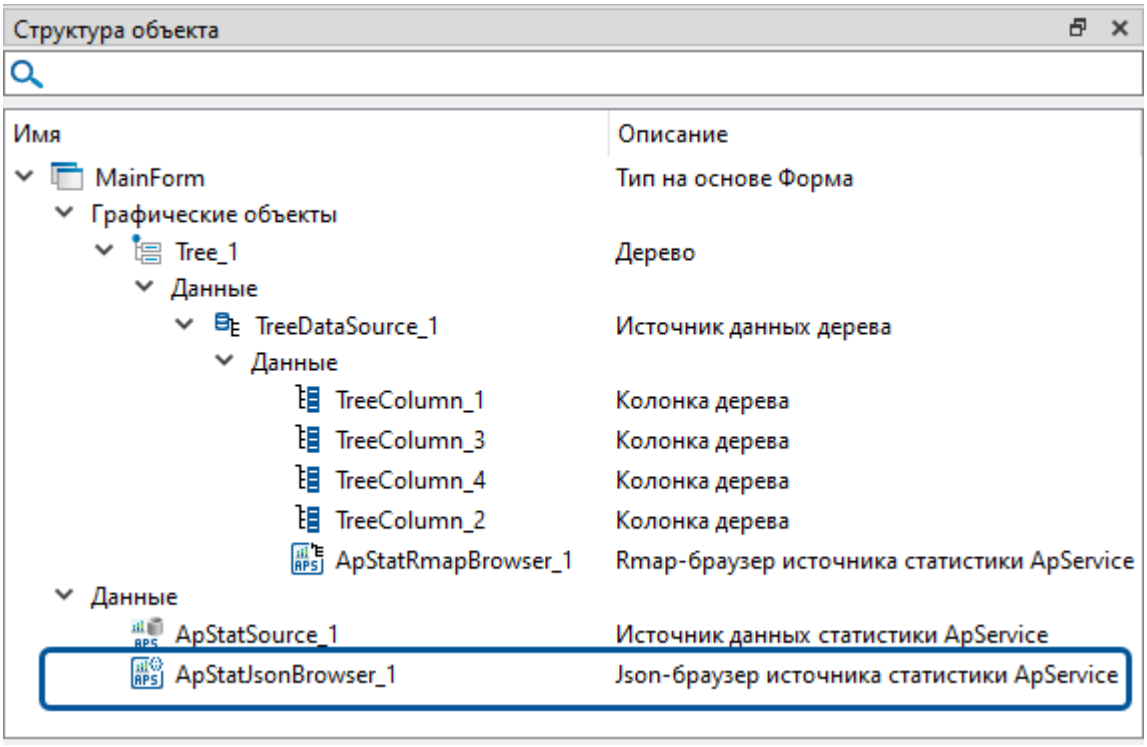


3. Для просмотра узлов дерева статистики сразу после открытия экранной формы в рантайме, воспользуйтесь обработчиком события Opened. Пропишите в нем следующий код:

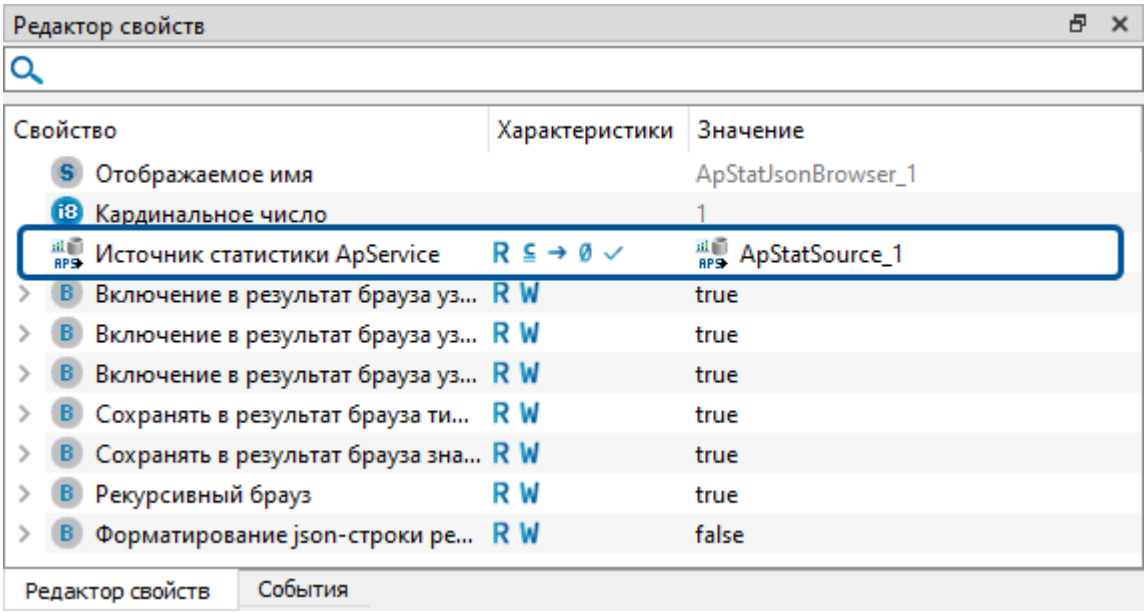
```
Tree_1.TreeDataSource_1.ApStatRmapBrowser_1.GetChildren("");
```

14.6.4. Сформировать срез дерева статистики с помощью компонента Json-браузер источника данных статистики ApService

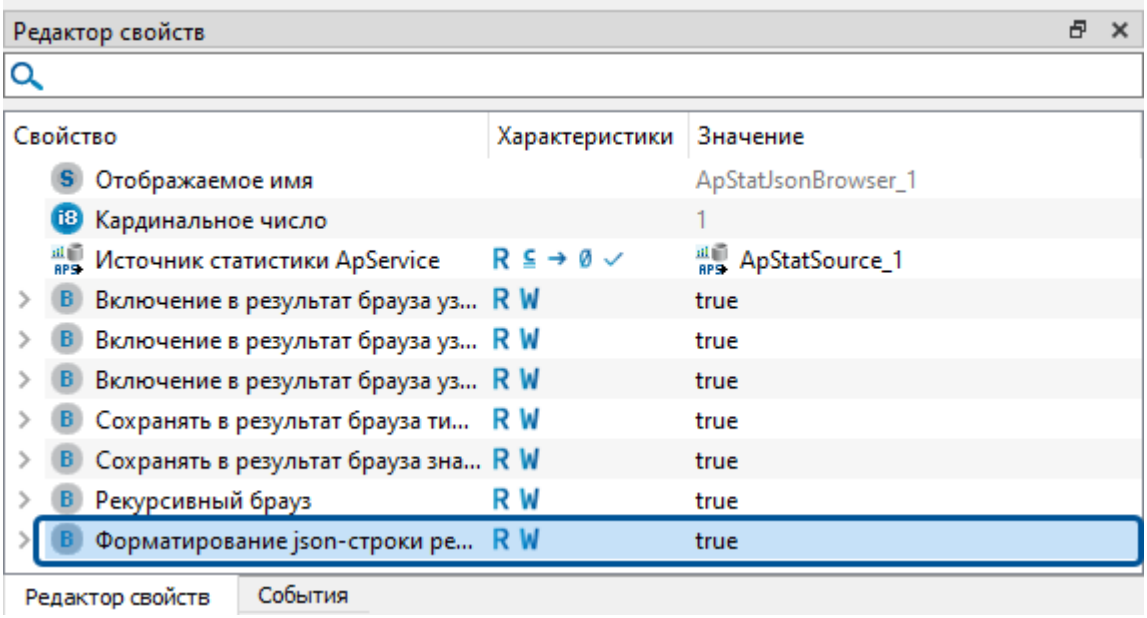
1. Чтобы сформировать срез дерева статистики в формате JSON, добавьте на форму компонент **`Json-браузер источника данных статистики ApService`**.



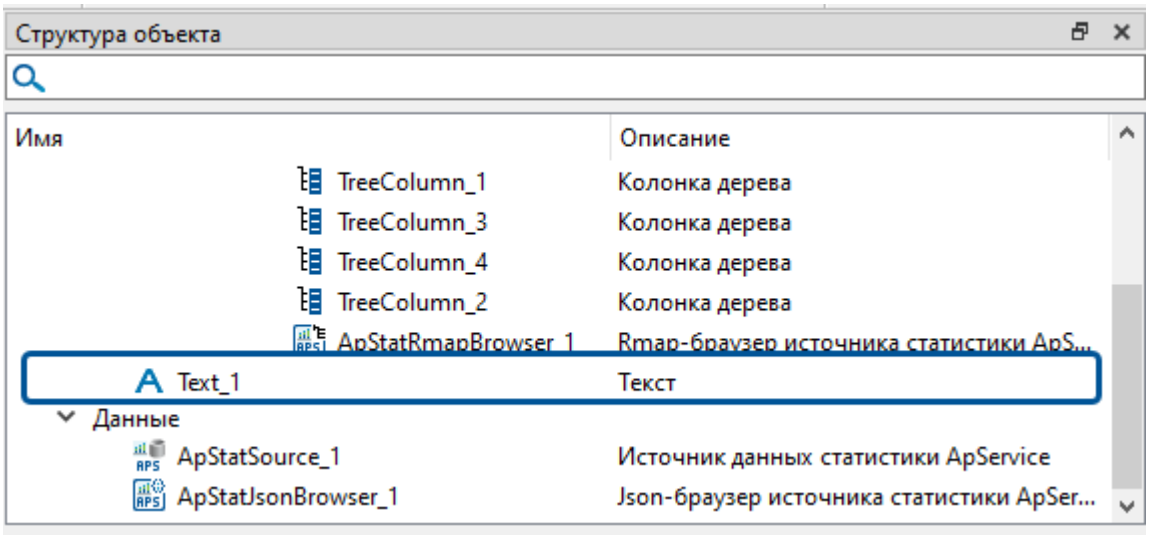
2. В редакторе свойств у объекта **`Json-браузер источника данных статистики ApService`** укажите ссылку на добавленный ранее компонент **`Источник данных статистики ApService`**. Для этого нажмите правой кнопкой мыши по свойству Источник статистики ApService → **`Сослаться`** → **`ApStatSource_1`**.



3. Для удобства просмотра среза дерева статистики с отступами и переносами строк, переведите свойство **Форматирование json-строки результата брауза в читабельном виде** в режим **«true»**.



4. Добавьте на форму текстовое поле, куда будет выводиться срез дерева статистики.



5. Для формирования среза дерева статистики сразу после открытия экранной формы в рантайме, воспользуйтесь обработчиком события Opened. Используйте метод GetChildren компонента **Json-браузер источника данных статистики ApService**, передав пустую строку в качестве аргумента, чтобы получить срез от корневого узла

```
ApStatJsonBrowser_1.GetChildren("");
```

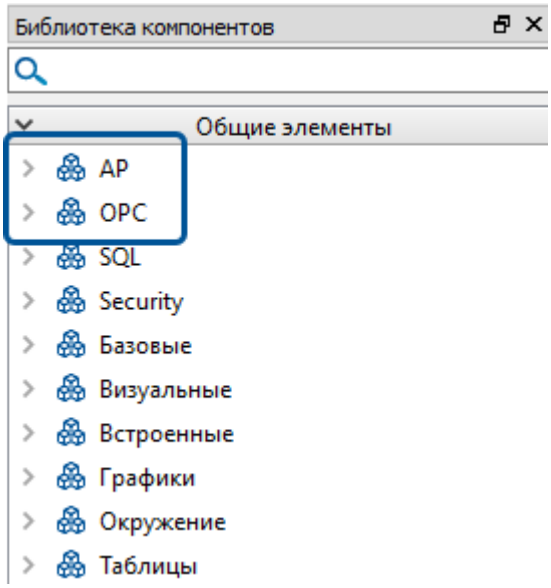
6. Чтобы получить результаты сформированного среза дерева статистики, в обработчике события GetChildrenComplete у компонента **Json-браузер источника статистики ApService** пропишите следующий код:

```
Text_1.Text = result;
```

15. Источники данных

В своем проекте вы можете обмениваться данными (запись и чтение) с любым сервером, работающим по протоколам OPC DA и TCP. Таким сервером может быть, к примеру, SePlatform.Data Server.

Все поставщики данных в SePlatform.HMI называются источниками данных. Для настройки подключения и взаимодействия с источниками данных существуют отдельные юниты в библиотеке компонентов: **AP** и **OPC**.



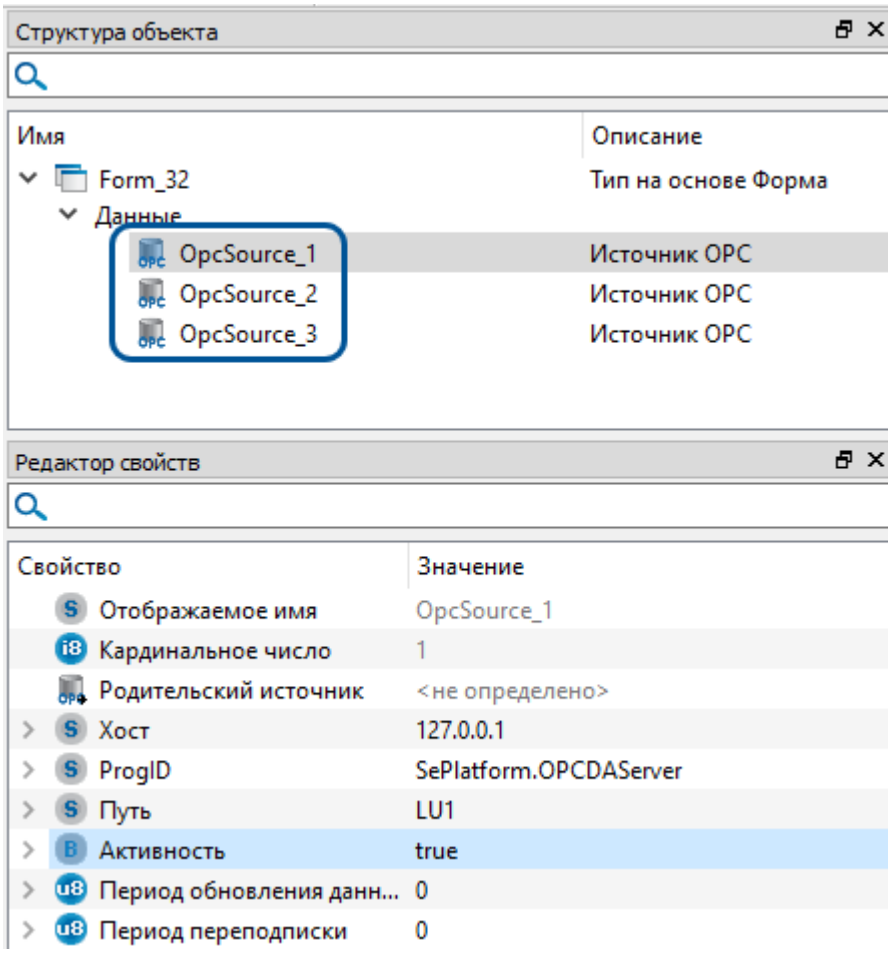
ОБРАТИТЕ ВНИМАНИЕ

Компоненты юнита **OPC** недоступны при работе в ОС Linux, т.к. спецификация OPC DA базируется на COM/DCOM и используется только в ОС Windows. Чтобы работать с источниками данных в ОС Linux, используйте компоненты юнита **AP**.

15.1. OPC DA

15.1.1. Подключение к источнику

Чтобы взаимодействовать с OPC DA источником данных, добавьте на экранную форму один или несколько компонентов **Источник OPC**. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на форму было добавлено 3 компонента **Источник OPC**.

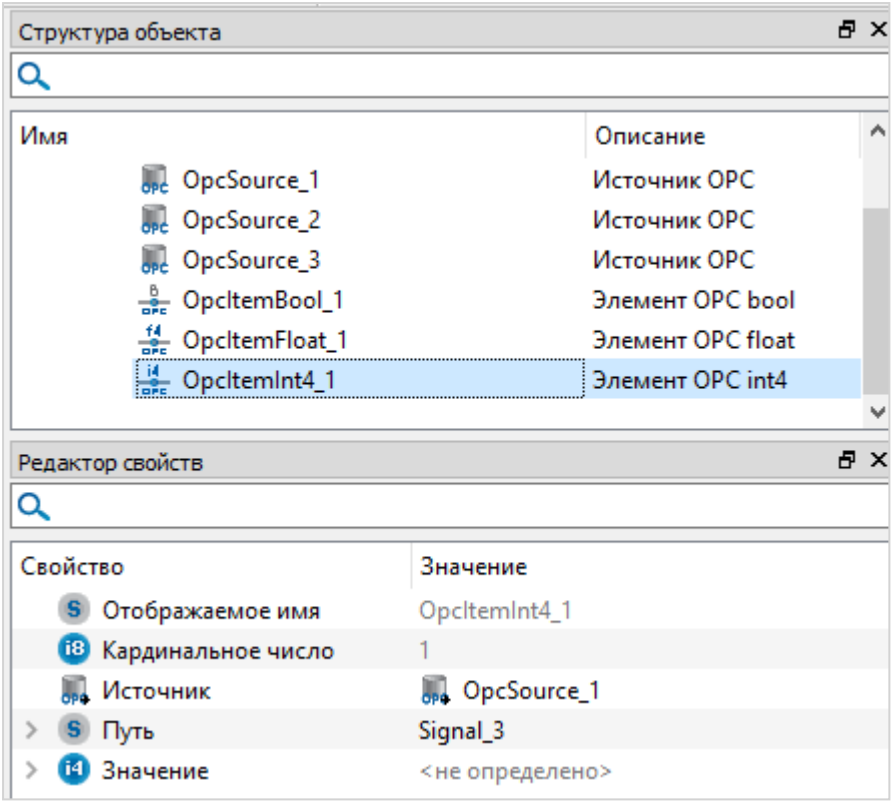


Чтобы компонент мог взаимодействовать с источником данных, настройте минимальный набор его свойств (стр. 1).

Свойство	Описание
Хост	Сетевой адрес OPC DA источника данных. Для локального компьютера используется IP-адрес: **«127.0.0.1»**.
ProgID	ProgID источника данных. Для SePlatform.Data Server данный параметр содержится в конфигурационном файле `SePlatform.Server.xml`. По умолчанию параметр равен **«SePlatform.OPCDAServer»**.
Путь	Путь до целевой ветки дерева сигналов OPC DA источника данных. Если путь не указан, то по умолчанию целевой веткой будет считаться корень дерева сигналов источника данных.
Активность	Активность источника OPC. С неактивными источниками не происходит обмен данными.

15.1.2. Работа с сигналами

Для обеспечения совместимости типов и оптимального распределения ресурсов при взаимодействии с сигналами источника данных, необходимо учитывать тип сигнала на сервере. Каждому типу сигнала на сервере соответствует отдельный компонент **Элемент OPC <T>**, где <T> - тип сигнала на сервере. Если тип сигнала на сервере неизвестен, используйте универсальный компонент **Элемент OPC**, который оперирует универсальным типом данных variant. После определения типов сигналов добавьте нужное количество компонентов **Элемент OPC <T>** на экранную форму из расчета один сигнал - один элемент OPC. Компонент - невизуальный и виден только в области **Структура объекта**.



Чтобы компонент **Элемент OPC <T>** мог взаимодействовать с конкретным сигналом источника данных, настройте минимальный набор его свойств (стр. 1).

Свойство	Описание
Источник	Ссылка на компонент **Источник OPC**, к дереву которого относится данный **Элемент OPC**.
Путь	Путь до сигнала относительно корня Источника.

15.2. TCP

Если ваш источник данных работает по протоколу TCP, вы можете получить от него оперативные и исторические значения, а также оперативные и исторические события.

ПРИМЕЧАНИЕ

Подробнее о том, как SePlatform.Data Server генерирует события, см. в руководстве администратора на модуль OPC AE Server. О том, как SePlatform.Data Server предоставляет данные по TCP - в руководстве администратора на модуль TCP Server Module.

15.2.1. Подключение к источнику

Чтобы взаимодействовать с источником данных по протоколу TCP, добавьте на экранную форму один или несколько компонентов **Источник AP**. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на форму было добавлено 3 компонента **Источник AP**.

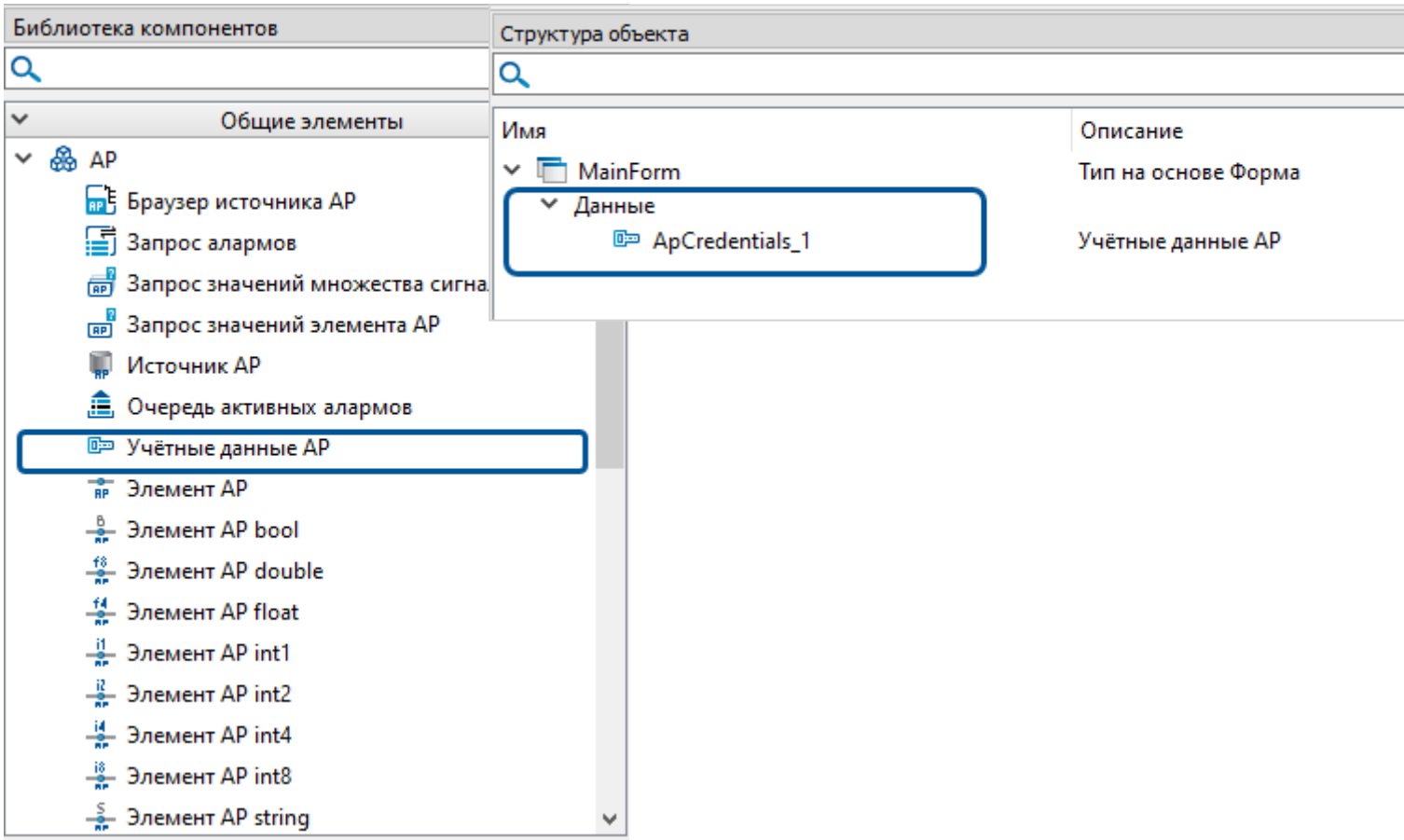
Чтобы компонент мог взаимодействовать с источником данных, настройте минимальный набор его свойств (стр. 1).

Свойство	Описание
Хост	Сетевой адрес источника данных по протоколу TCP. Для локального компьютера используется IP-адрес: **«127.0.0.1»**.
Порт	Номер порта для подключения к источнику данных. По умолчанию номер **«4388»**.
Путь	Путь до целевой ветки дерева сигналов источника данных по протоколу TCP. Если путь не указан, то по умолчанию целевой веткой будет считаться корень дерева сигналов источника данных.
Активность	Активность источника AP. С неактивными источниками не происходит обмен данными.

15.2.2. Подключение к источнику, защищенному паролем

Для подключения к источнику данных, защищенному паролем, такому как SePlatform.Data Server или аналогичному, используйте компонент **Учетные данные AP**. Компонент хранит ваши данные в виде хэша, обеспечивая их безопасность. При подключении к защищенному источнику, компонент автоматически загружает и использует хэшированный пароль, который используется для аутентификации и получения безопасного доступа к данным.

Компонент **Учётные данные AP** расположен в юните **«AP»Библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



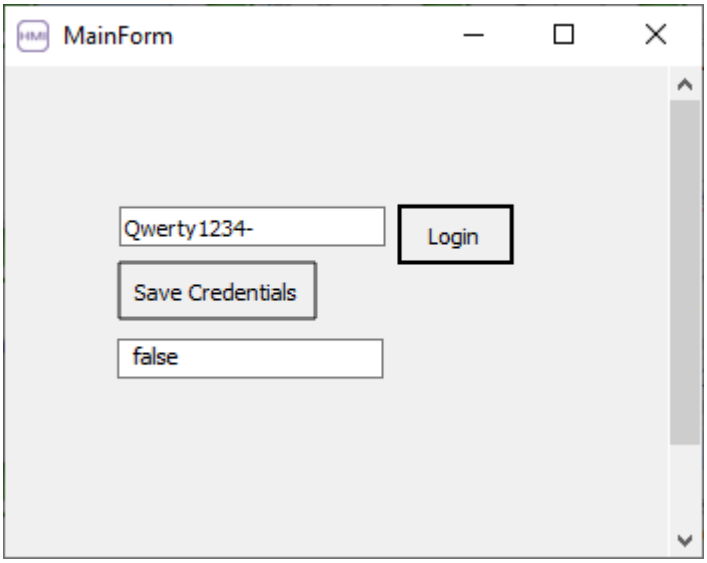
15.2.3. Как получить данные с источника, защищенного паролем (Демо-проект)

Далее кратко описан небольшой пример (файлы проекта идут в комплекте с документацией) использования компонента **Учетные данные АР** в проекте SePlatform.HMI с использованием языка SePlatform.Om.

В примере показано, как:

- Настроить подключение к SePlatform.Data Server, защищенному паролем.
- Добавить кнопку для аутентификации и получения доступа к защищенному источнику.
- Настроить автоматическую аутентификацию к защищенному источнику.
- Получить текущее значение сигнала с защищенного источника данных после успешной аутентификации.

В результате будет подготовлена форма, позволяющая вводить пароль от защищенного источника данных и после успешной аутентификации получать текущее значение сигнала.

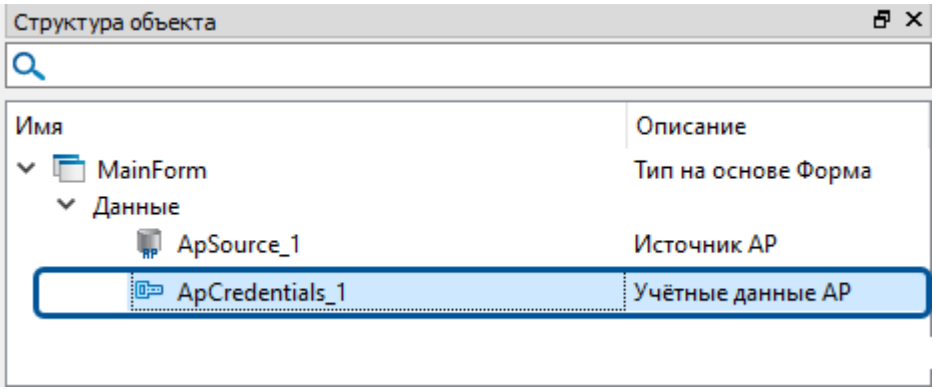


ПРИМЕЧАНИЕ

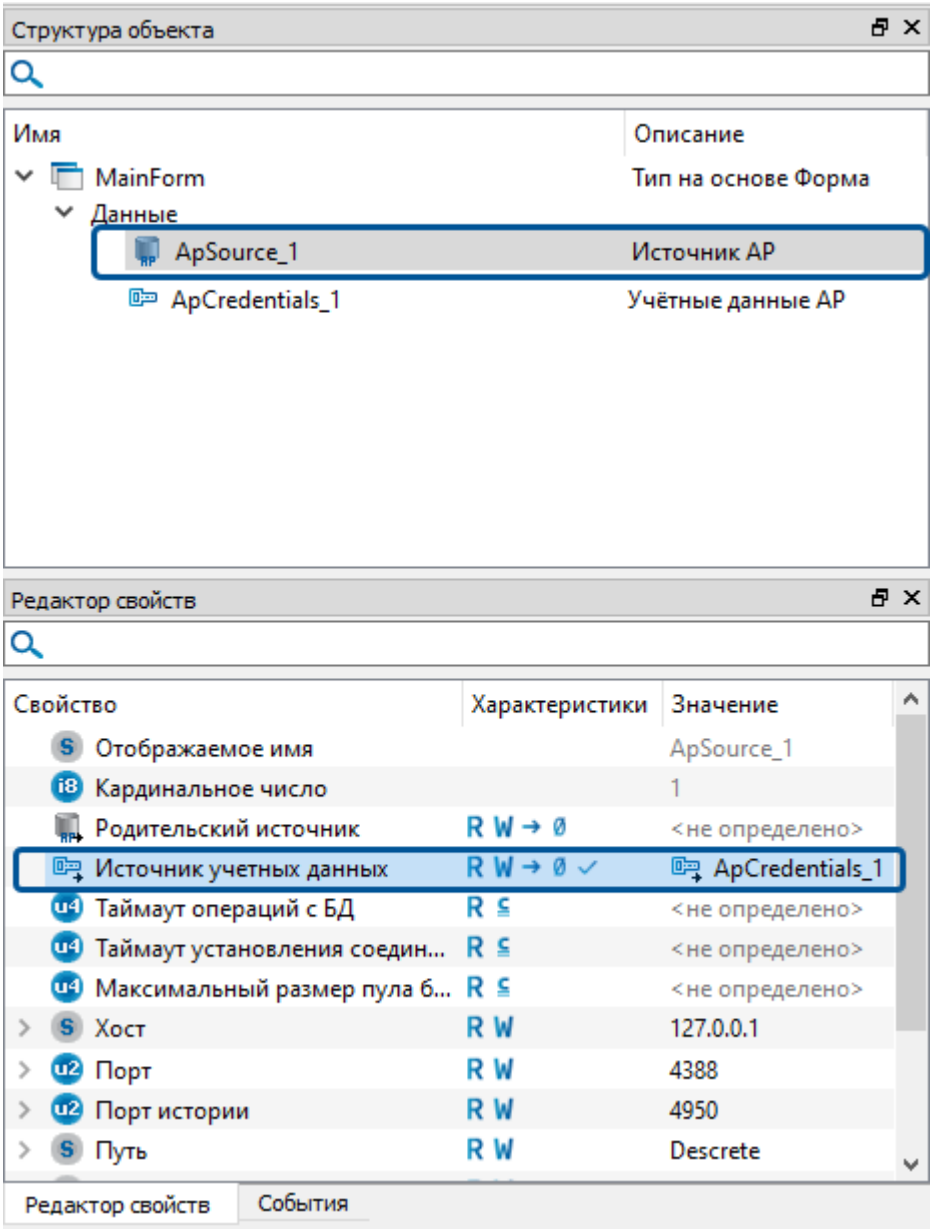
Для работы с проектом-примером в проект уже должен быть добавлен объект типа **Источник АР**, с которым вы планируете взаимодействовать. Для получения подробной информации о настройке и подключении источника данных обратитесь к разделу Подключение к источнику (стр. 147).

15.2.3.1. Настроить подключение к защищенному источнику данных

1. Чтобы подключиться к SePlatform.Data Server, который защищен паролем, добавьте на форму компонент **Учетные данные АР.**

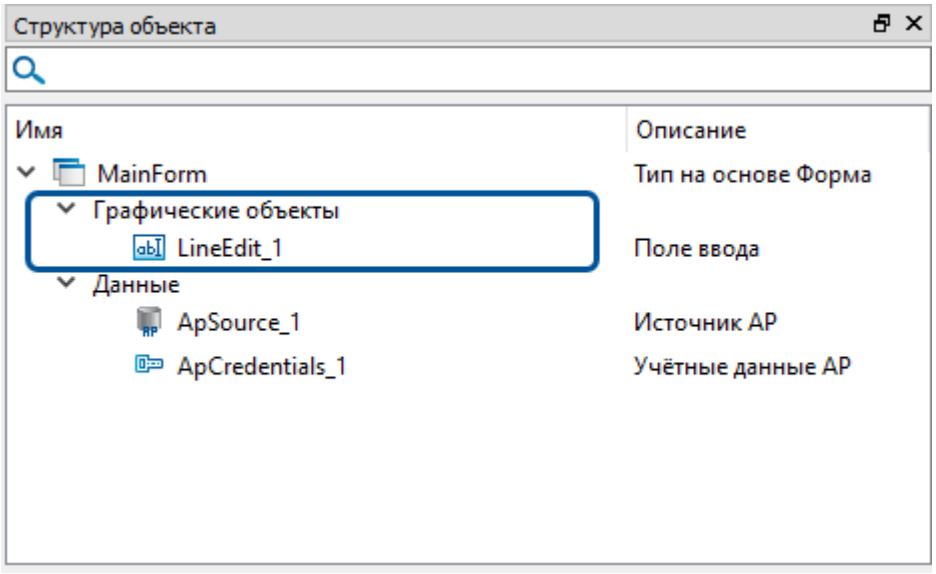


2. В структуре объекта выделите источник **Источник AP** и в свойстве Источник учетных данных сошлитесь на **Учетные данные AP**. Для этого нажмите правой кнопкой мыши по свойству Источник учетных данных → **Сослаться** → **ApCredentials_1**.

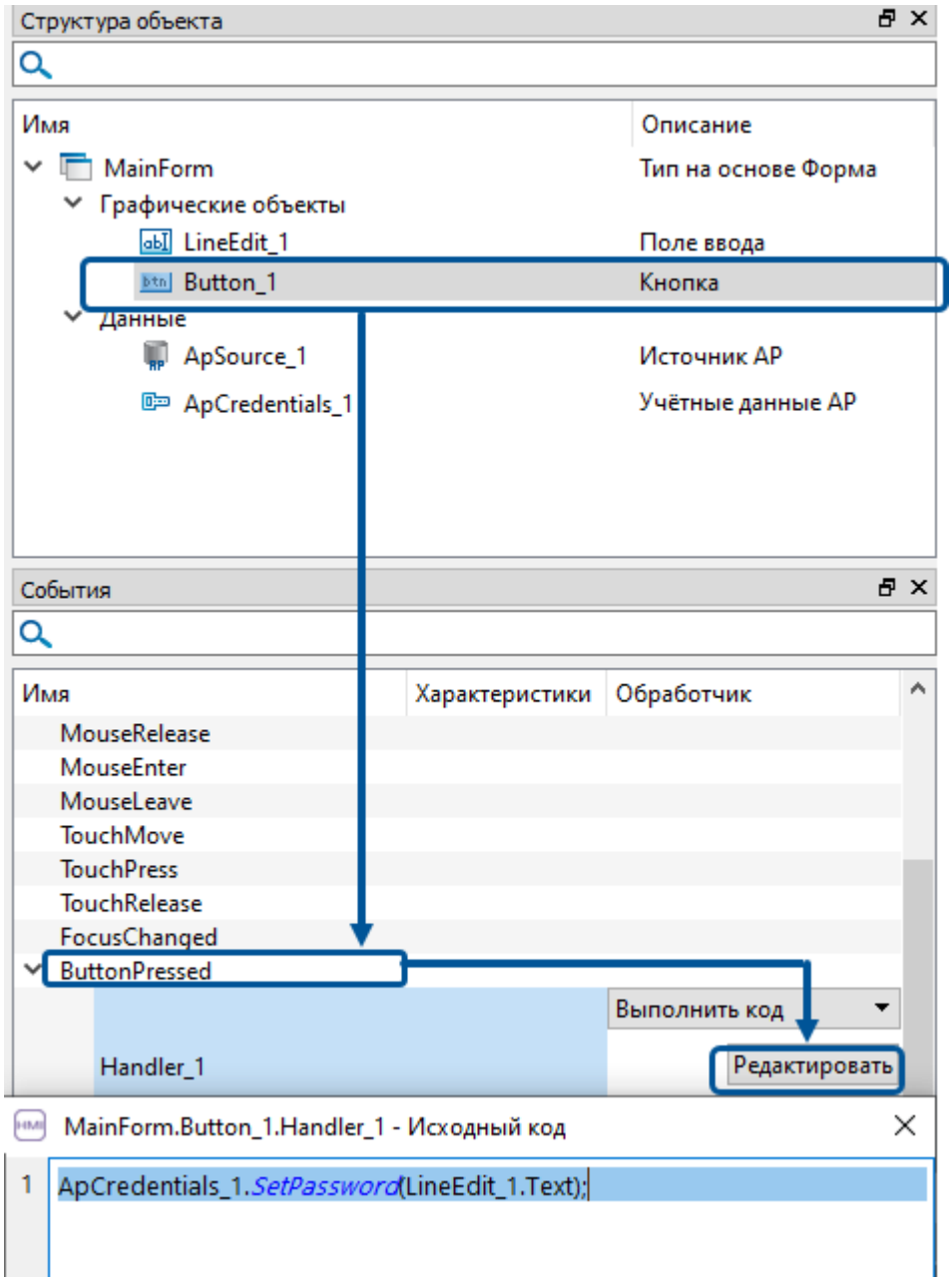


15.2.3.2. Добавить кнопку для аутентификации и получения доступа к защищенному источнику

1. Чтобы по нажатию кнопки подключаться к защищенному источнику, подготовьте поле ввода, где будет вводиться пароль доступа к SePlatform.Data Server.



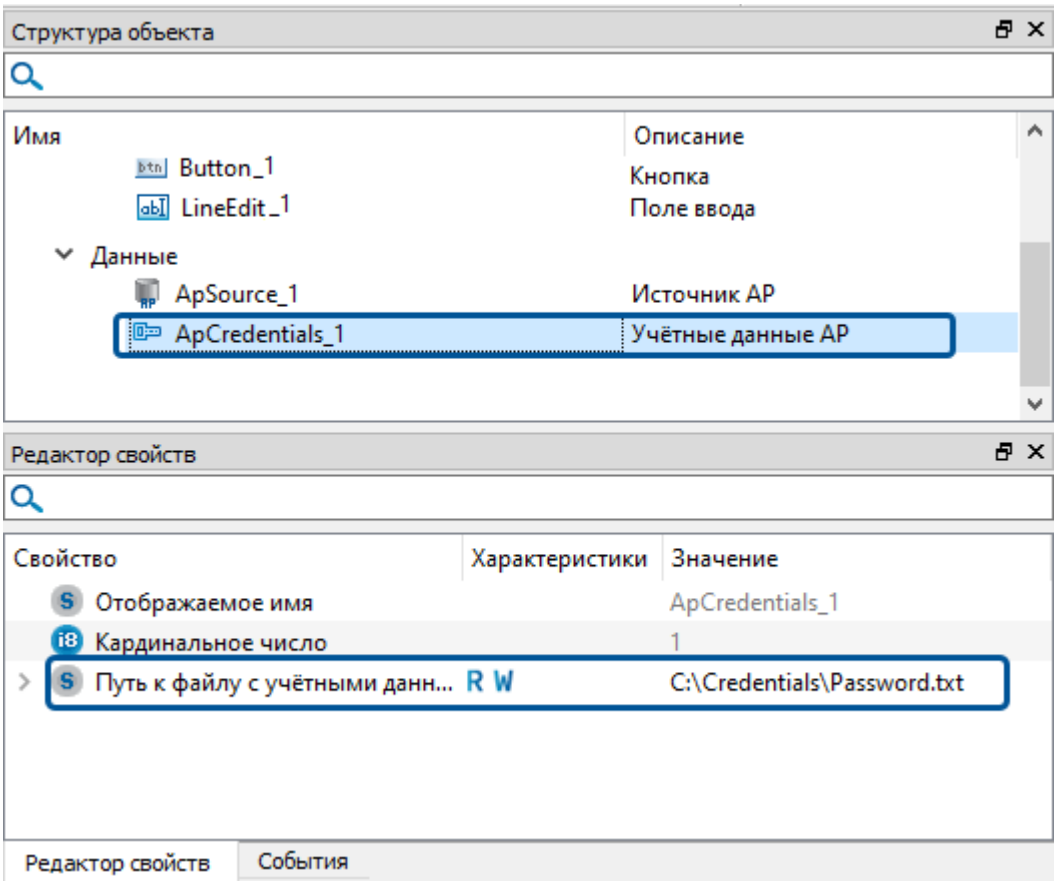
2. Добавьте на форму компонент **Кнопка**по нажатию которой будет происходить аутентификация для подключения к защищенному источнику. В обработчике ButtonPressed пропишите следующий код:



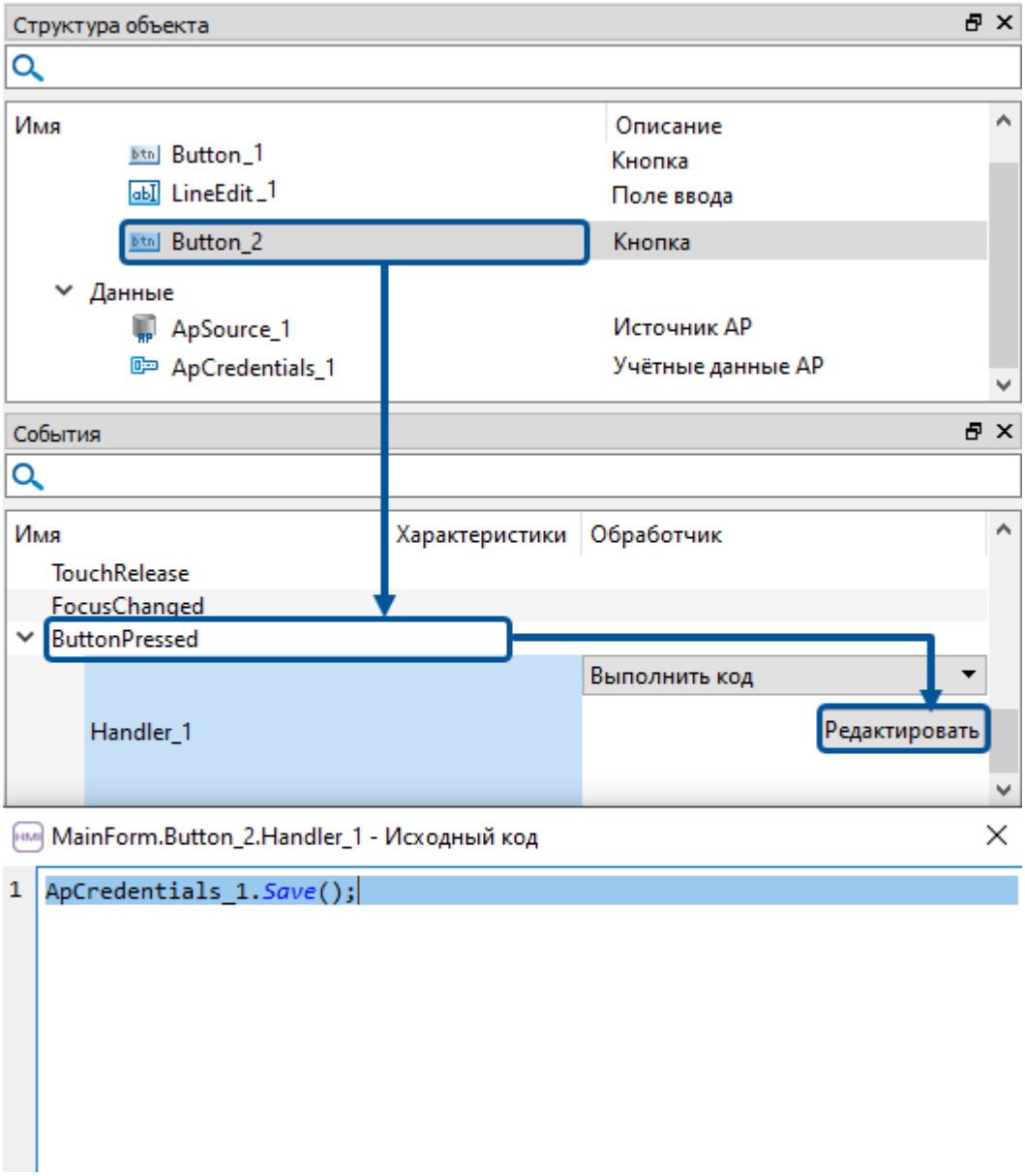

```
ApCredentials_1.SetPassword(LineEdit_1.Text)
```

15.2.3.3. Настроить автоматическую аутентификацию к защищенному источнику

1. Для того чтобы при каждом запуске проекта в рантайме не вводить повторно пароль, в свойстве Путь к файлу с учетными данными укажите путь к файлу, где будет храниться хэш пароля, который будет использоваться для автоматической аутентификации и получения доступа к защищенному источнику.


Структура объекта
Имя
Описание
Button_1
Кнопка
LineEdit_1
Поле ввода
Данные
ApSource_1
Источник AP
ApCredentials_1
Учётные данные AP
Редактор свойств
Свойство
Характеристики
Значение
Отображаемое имя
ApCredentials_1
Кардинальное число
1
Путь к файлу с учётными данн...
R W
C:\Credentials\Password.txt
Редактор свойств
События

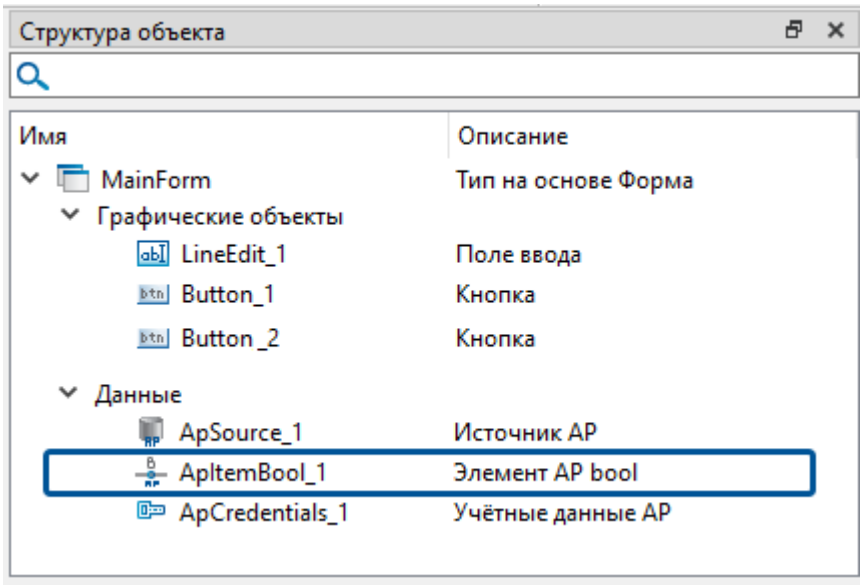
2. Добавьте на форму компонент **Кнопка**по нажатию которой будет сохраняться хэш пароля в указанный файл. В обработчике ButtonPressed пропишите следующий код:



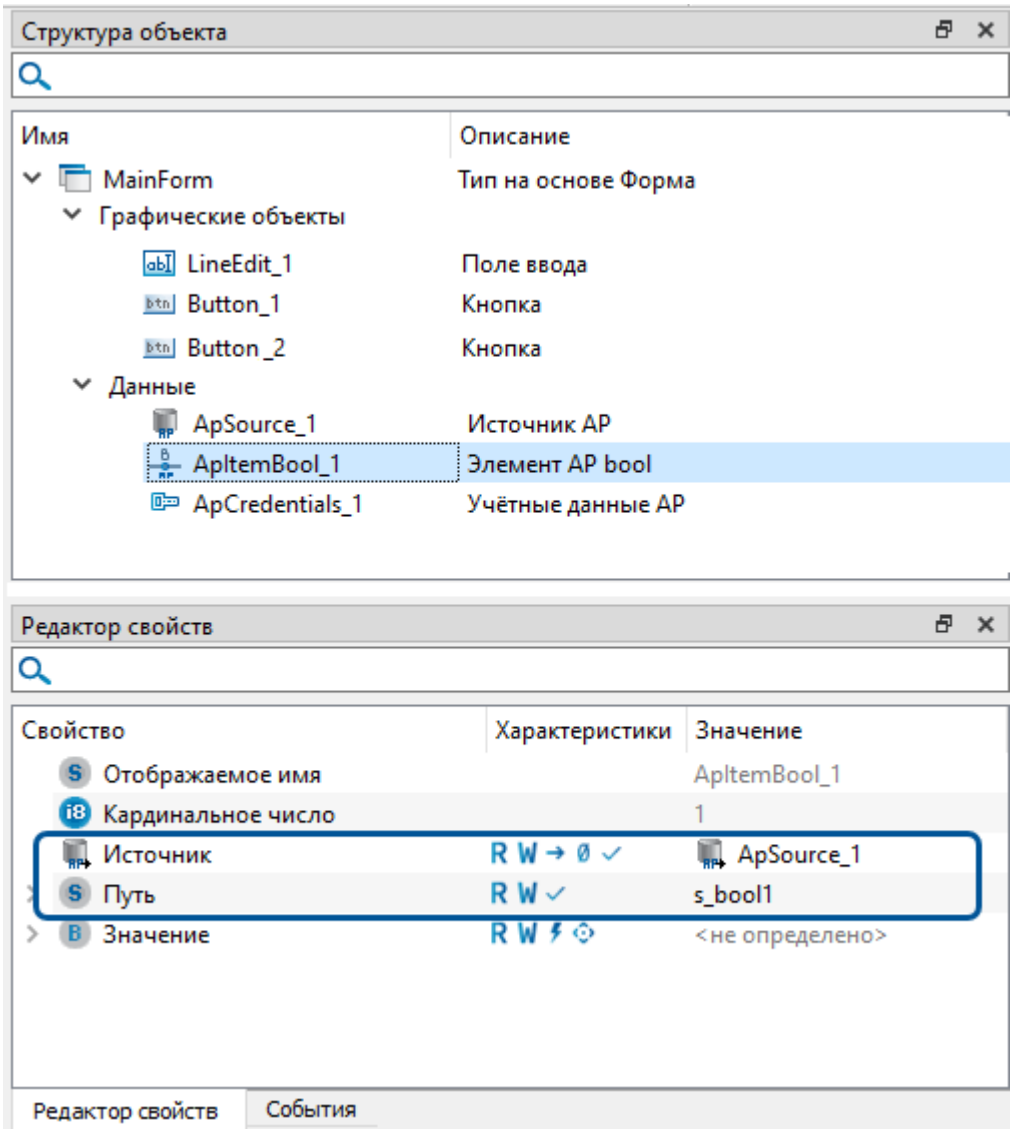

```
ApCredentials_1.Save();
```

15.2.3.4. Получить текущее значение сигнала с защищенного источника данных

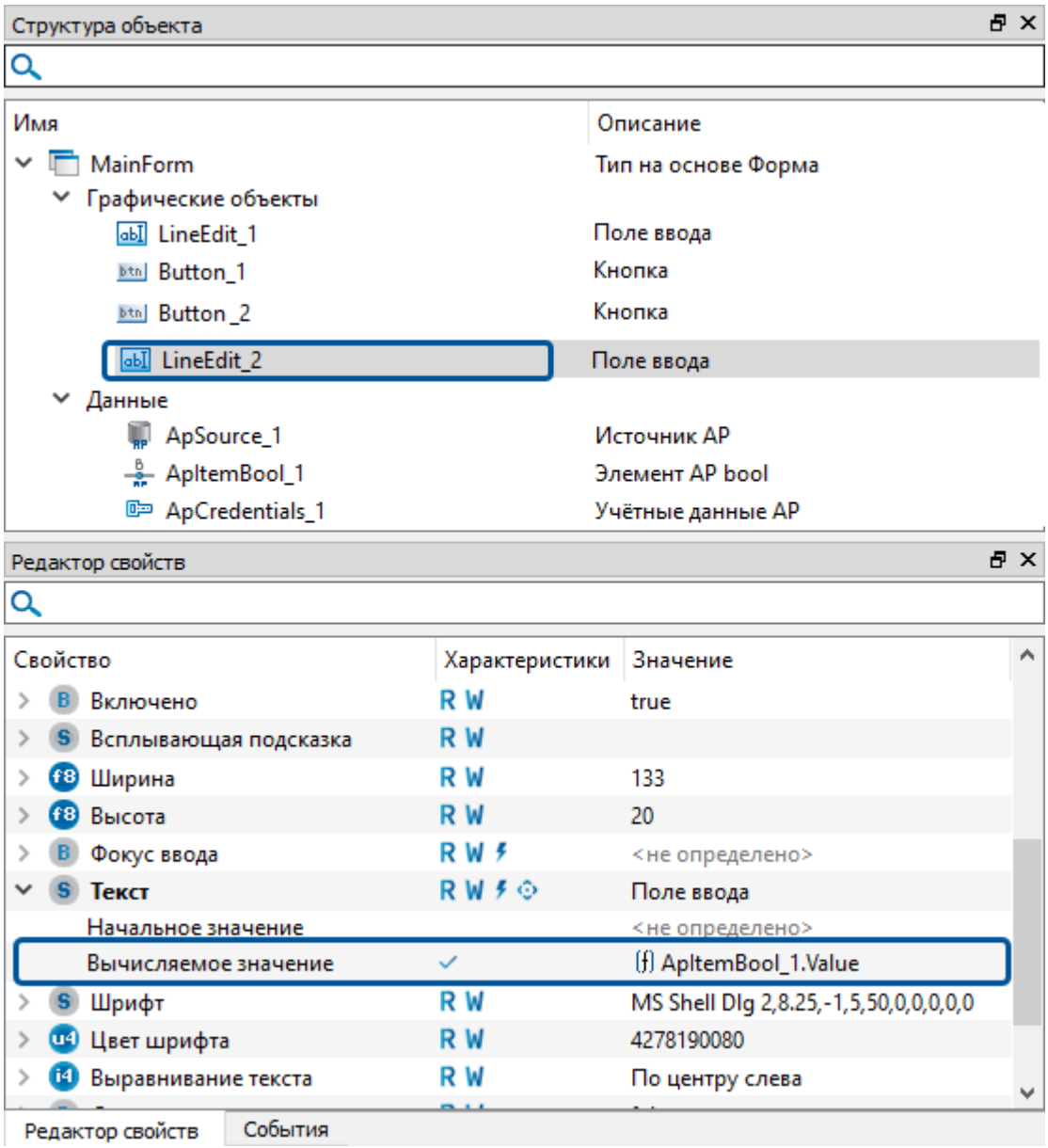
1. Чтобы выводить текущее значение сигнала с защищенного источника данных на экранную форму добавляются элементы **«АР»** - набор компонентов для взаимодействия с сигналами определенного типа. Например, для работы с булевым сигналом, добавьте на экранную форму компонент **Элемент АР bool**.



2. В свойстве Источник (стр. 1) укажите ссылку на компонент **Источник AP**, к дереву которого относится данный **Элемент AP**. Укажите путь до сигнала относительно корня Источника с помощью свойства Путь (стр. 1).



3. Добавьте на форму **Поле ввода**, чтобы выводить текущее значение сигнала источника данных. В свойстве Текст задайте вычисляемое значение по формуле:




```
ApItemBool_1.Value
```

После запуска формы в рантайме в поле ввода отобразится текущее значение сигнала «**s_bool1**».

Возможные причины, по которым значение может не отобразиться:

- Неудачная аутентификация. Если значение не отображается, возможно, произошла неудачная аутентификация. Рекомендуется перепроверить правильность введенного пароля.
- Отсутствие соединения с источником. Если значение не отображается, убедитесь, что свойство Активность (стр. 1) источника данных включено.
- Некорректные настройки компонентов. Неправильные настройки компонентов, такие как неправильно указанный путь к сигналу в свойстве Путь (стр. 1) может привести к отсутствию отображения значения сигнала.

15.2.4. Работа с сигналами

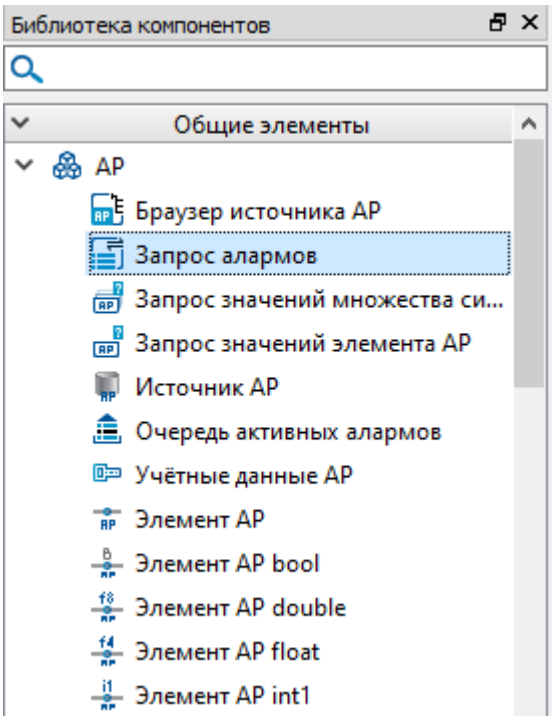
Для обеспечения совместимости типов и оптимального распределения ресурсов при взаимодействии с сигналами источника данных, необходимо учитывать тип сигнала на сервере. Каждому типу сигнала на сервере соответствует отдельный компонент **Элемент AP <T>**, где <T> - тип сигнала на сервере. Если тип сигнала на сервере не известен, используйте универсальный компонент **Элемент AP**, который оперирует универсальным типом данных variant. После определения типов сигналов добавьте нужное количество компонентов **Элемент AP <T>** на экранную форму из расчета один сигнал - один элемент AP. Компонент - невизуальный и виден только в области **Структура объекта**.

Чтобы компонент **Элемент AP <T>** мог взаимодействовать с конкретным сигналом источника данных, настройте минимальный набор его свойств (стр. 1).

Свойство	Описание
Источник	Ссылка на компонент **Источник AP**, к дереву которого относится данный **Элемент AP**.
Путь	Путь до сигнала относительно корня Источника.

15.2.5. Получение событий

Чтобы запросить с источника оперативные или исторические события, используйте компонент **Запрос алармов**.



Компонент вычитывает все события (даже квитированные и деактивированные), но вы можете установить фильтры в свойствах компонента (описание свойств, функций и событий компонента см. в справочном руководстве) и запрашивать только те события, которые вас интересуют.

ПРИМЕЧАНИЕ

О событиях и их возможных состояниях см. в руководстве администратора на модуль OPC AE Server.

Как настроить запрос событий

Чтобы компонент **Запрос алармов** подключался к серверу и запрашивал все события (включая квитированные и деактивированные):

1. Добавьте объект типа **Запрос алармов** на форму.
2. Укажите источник данных - сервер, с которого вы хотите запрашивать события, в свойстве **Источник**.

   ПРИМЕЧАНИЕ

   В проект уже должен быть добавлен объект типа **Источник АР**.

3. Укажите какие именно данные и в каком виде вы хотите получать: оперативные события журналом, исторические события журналом или список активных условий. Используйте для этого свойство **Режим работы**.
4. Укажите хотите ли вы получать все существующие активные события или только те, которые возникли после подключения компонента к серверу. Используйте для этого свойство **Запрашивать список активных событий при подключении к источнику**.
5. Активируйте компонент в свойстве **Активность**.
6. Чтобы обновлять данные от источника, используйте функцию **Reload**.

Чтобы среди запрошенных событий не было квитированных и деактивированных, установите **«true»** в свойстве **Удалять квитированные и деактивированные события**.

Какие данные получает компонент

По каждому событию компонент **Запрос алармов** получает набор данных.

Идентификатор	**Описание**	**Название столбца по умолчанию (если не заполнять свойство Заголовок для компонента Таблица: столбец)**
«source»	Источник события (полный тег сигнала, изменение которого привело к генерации события).	Источник
«time»	Время генерации уведомления о событии.	Время генерации
«message»	Сообщение.	Сообщение
«severity»	Уровень важности события.	Важность
«condition_name»	Имя условия генерации события.	Условие
«subcondition_name»	Имя подусловия генерации события.	Подусловие

Идентификатор	Описание	Название столбца по умолчанию (если не заполнять свойство Заголовок для компонента Таблица: столбец)
«quality»	Текущее качество сигнала, изменение которого привело к генерации события.	Качество
«active_time»	Время перехода состояния события в активное.	Время срабатывания
«actor_id»	Имя пользователя, выполнившего квитирование сообщения о событии.	Пользователь
«ack»	Признак квитирования события.	Квитировано
«active»	Признак активности подусловия, по которому было сгенерировано событие.	Активность
«cookie»	Специальный идентификатор события, который имеет служебное назначение.	Куки
«ack_time»	Время квитирования.	Время квитирования
«ack_required»	Требование квитирования.	Квитировать
«value»	Значение сигнала, изменение которого привело к генерации события.	Значение
«sound»	Звуковой файл или группа звуковых файлов, воспроизводимых при выполнении подусловия генерации сообщения о событии.	Звук
«area_path»	Тег родительского узла источника события.	Зона
«object_id»	Идентификатор объекта на сервере, по которому было сгенерировано событие.	ID объекта
«object_name»	Имя объекта на сервере, по которому было сгенерировано событие.	Имя объекта
«relative_tag»	Относительный тег источника события. Состоит из имени объекта («object_name») и той части полного тега источника события, которая находится после имени объекта.	Относительный тег
«deactive_time»	Время, когда событие перешло из активного состояния в неактивное.	Время деактивации

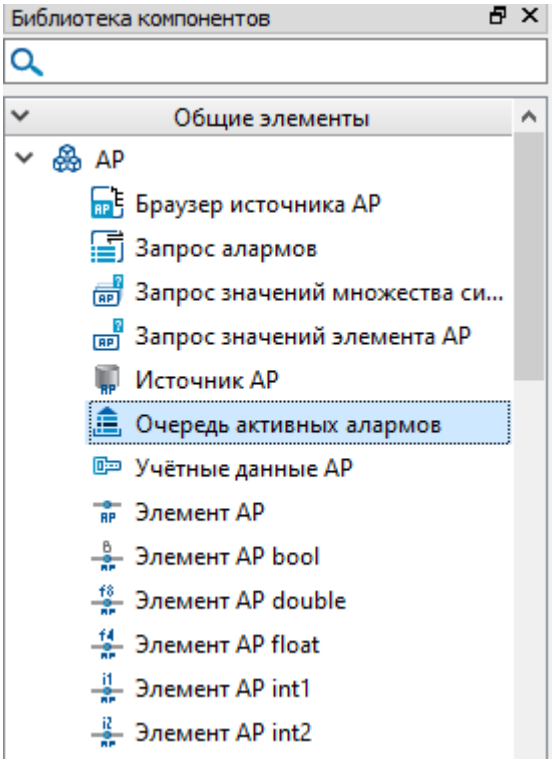
Используя указанные выше идентификаторы, вы можете обращаться к этим данным в проекте. Например, вывести их значения в визуальную таблицу на форме, используя компоненты **Таблица: модель данных** и **Таблица** из юнита **Таблицы**.

ПРИМЕЧАНИЕ

Чтобы юнит **Таблицы** появился в библиотеке компонентов, установите SePlatform.HMI.Tables.

Очередь активных событий

События, получаемые с помощью компонента **Запрос алармов** в оперативном режиме, можно помещать в очередь с помощью компонента **Очередь активных алармов**.



Очередь может содержать только неквитированные события. Квитированные события из очереди удаляются. Так вы можете использовать очередь, например, для построения очереди звуков неквитированных событий.

Чтобы работать с элементами очереди событий, используйте API компонента **Очередь активных алармов** (описание свойств, функций и событий компонента см. в справочном руководстве).

Как создать очередь активных событий

Чтобы организовать очередь из всех активных событий, сгенерированных сервером:

1. Добавьте компонент **Очередь алармов** на форму.
2. Укажите источник событий для очереди - объект типа **Запрос алармов** в свойстве Запрос алармов.

> ПРИМЕЧАНИЕ
> В проект уже должен быть добавлен объект типа **Запрос алармов**.

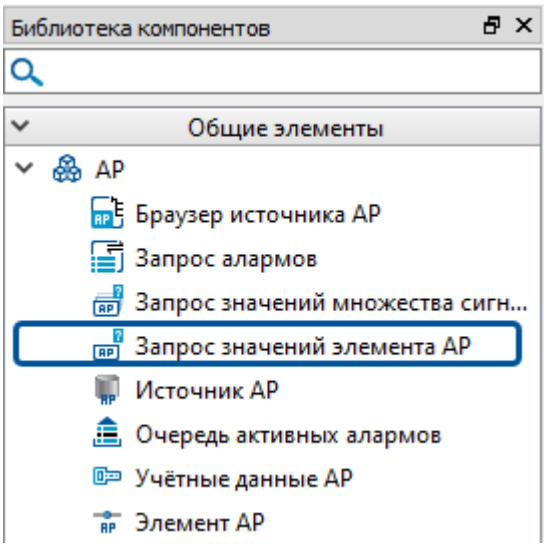
3. Чтобы обновлять данные от источника, используйте функцию Reload.

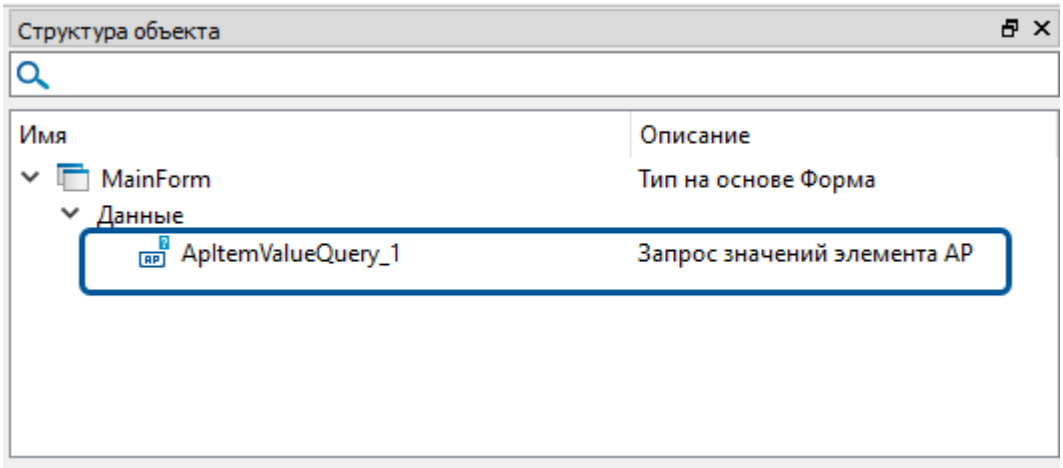
15.2.6. Получение значений одного сигнала

Чтобы запросить оперативные или исторические значения сигнала с источника данных (например, с SePlatform.Data Server) воспользуйтесь компонентом **Запрос значений элемента AP**. Компонент **Запрос значений элемента AP** работает с одним экземпляром компонента **Элемент AP**, обрабатывая данные только для одного сигнала. Компонент выполняет запрос к источнику и возвращает полученные значения сигнала

для дальнейшего использования в проекте SePlatform.HMI. Чтобы узнать подробности о возвращаемых значениях, обратитесь к разделу Какие данные получает компонент (стр. 161).

Компонент **Запрос значений элемента АР** расположен в юните **АР Библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.





Свойства и методы компонента **Запрос значений элемента АР** рассмотрены в справочном руководстве.

15.2.6.1. Какие данные получает компонент

Компонент **Запрос значений элемента АР** получает значения свойств сигнала:

Идентификатор	**Описание**	**Название столбца по умолчанию (если не заполнять свойство Заголовок для компонента Таблица: столбец)**
«value»	Значение сигнала	Значение
«quality»	Текущее качество сигнала	Качество
«timestamp»	Метка времени значения сигнала	Метка времени

Идентификатор	Описание	Название столбца по умолчанию (если не заполнять свойство Заголовок для компонента Таблица: столбец)
«value»	Значение сигнала	Значение
«server_timestamp»	Метка времени сервера, когда значение было получено. Работает только в историческом режиме.	Метка времени сервера

Используя указанные выше идентификаторы, вы можете обращаться к этим данным в проекте. Например, вывести их значения в визуальную таблицу на форме, используя компоненты **Таблица: модель данных** и **Таблица** из юнита **Таблицы**.

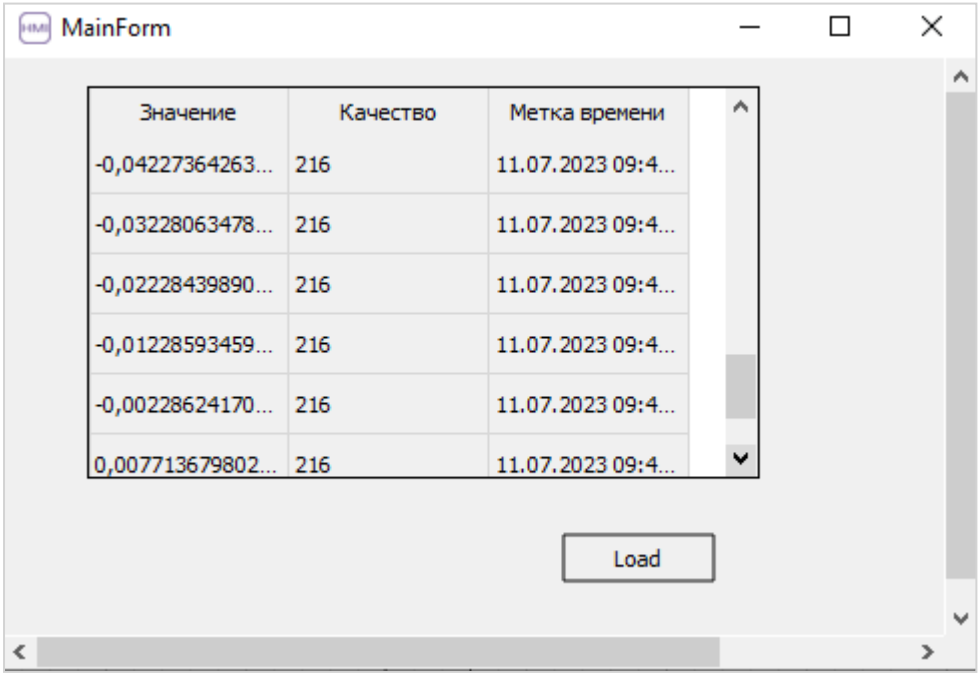
15.2.7. Как настроить запрос значений (Демо-проект)

Далее кратко описан небольшой пример (файлы проекта идут в комплекте с документацией) использования компонента **Запрос значений элемента АР** в проекте SePlatform.HMI (проект сделан на языке SePlatform.Om).

В примере показано, как:

- Настроить взаимодействие компонента **Запрос значений элемента АР** с источником данных для получения оперативных и исторических значений сигнала.
- Настроить вывод в таблицу значения сигнала, метки времени и качества сигнала.
- Добавить кнопку запроса данных с источника, поставляющего оперативные или исторические значения.

В результате будет подготовлена форма, на которой при нажатии на кнопку **Load** оперативные значения сигнала будут получены с SePlatform.Data Server и помещены в таблицу.

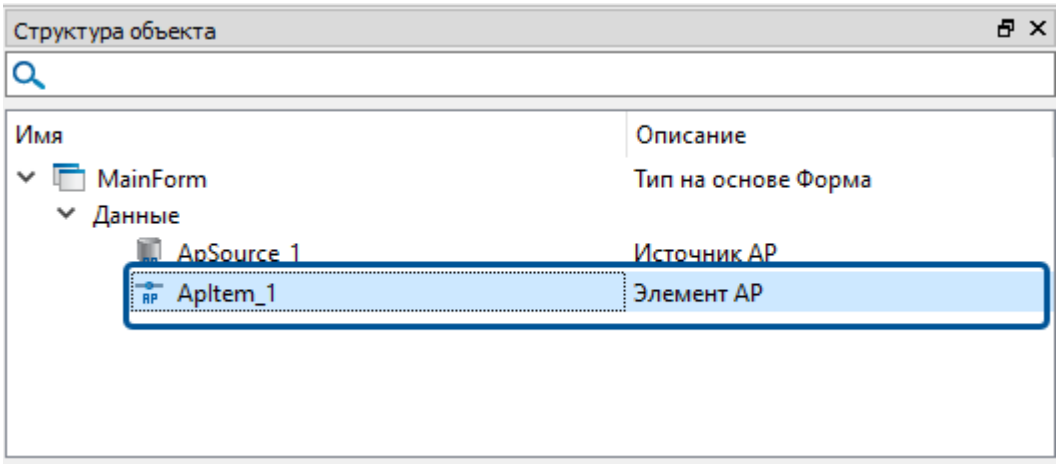


ПРИМЕЧАНИЕ

Убедитесь, что у вас уже настроено подключение к источнику данных, с которым вы планируете взаимодействовать. Для получения подробной информации о настройке и подключении источника данных обратитесь к разделу Подключение к источнику (стр. 147).

15.2.7.1. Настроить взаимодействие компонента Запрос значений элемента AP с источником данных

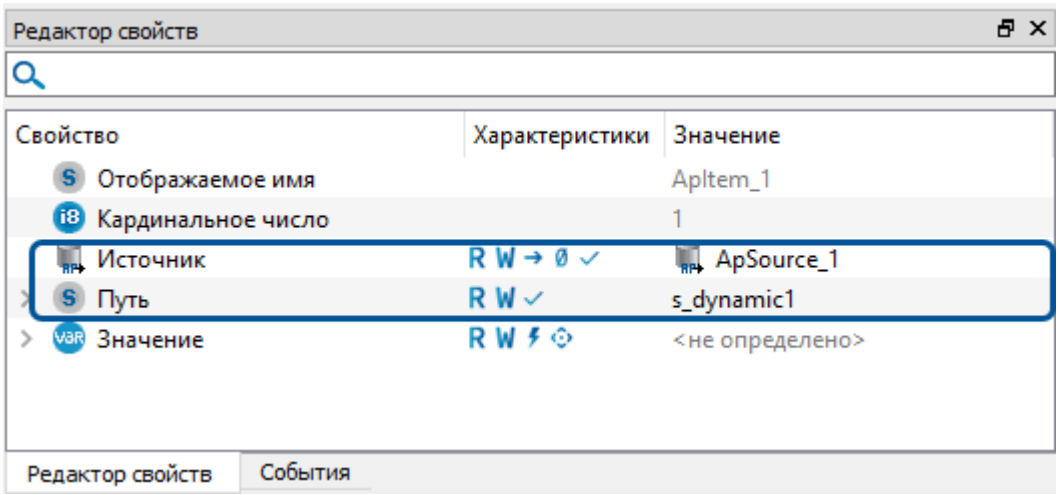
1. Добавьте на форму компонент **Элемент AP**, с которого компонент **Запрос значений элемента AP** будет запрашивать оперативные значения сигнала.



ОБРАТИТЕ ВНИМАНИЕ

Компонент **Запрос значений элемента AP** работает только с типом данных variant (т.е нетипизированным).

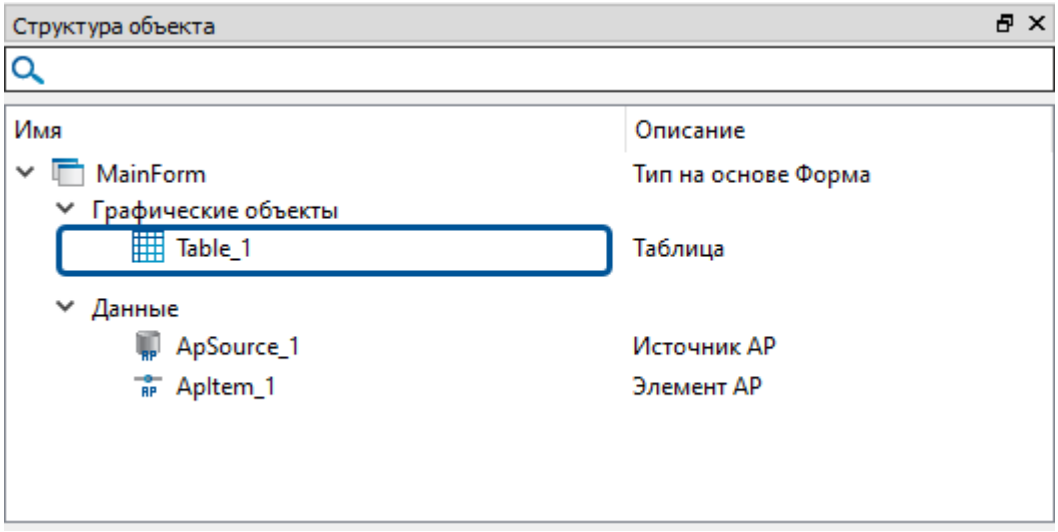
2. В свойстве **Источник** укажите ссылку на компонент **Источник AP**, к дереву которого относится данный **Элемент AP**. Укажите путь до сигнала относительно корня Источника с помощью свойства **Путь**.



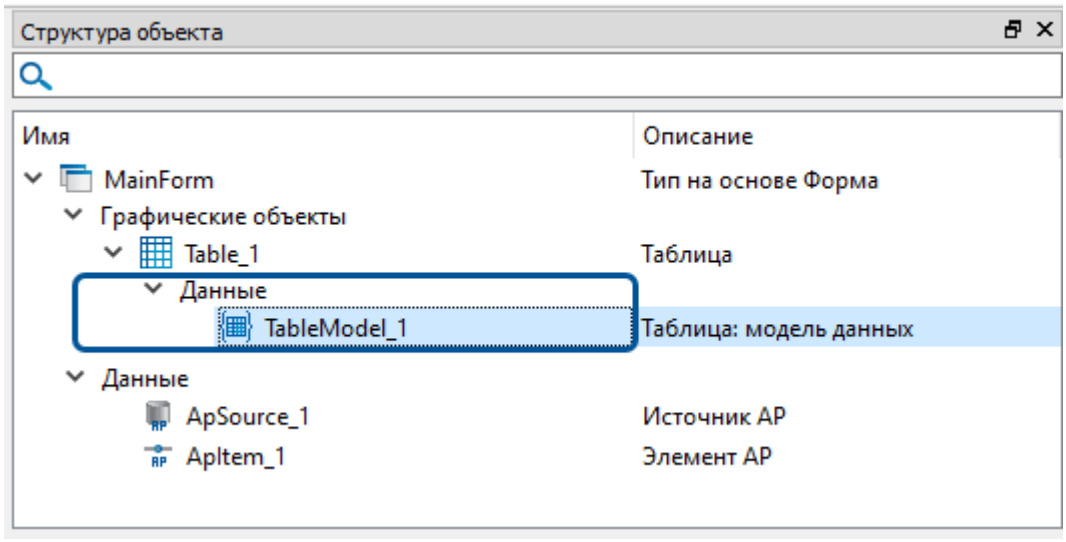
15.2.7.2. Настроить вывод в таблицу значения сигнала, метки времени и качества сигнала

1. Добавьте на экранную форму объект типа **Таблица**, чтобы иметь возможность помещать данные,

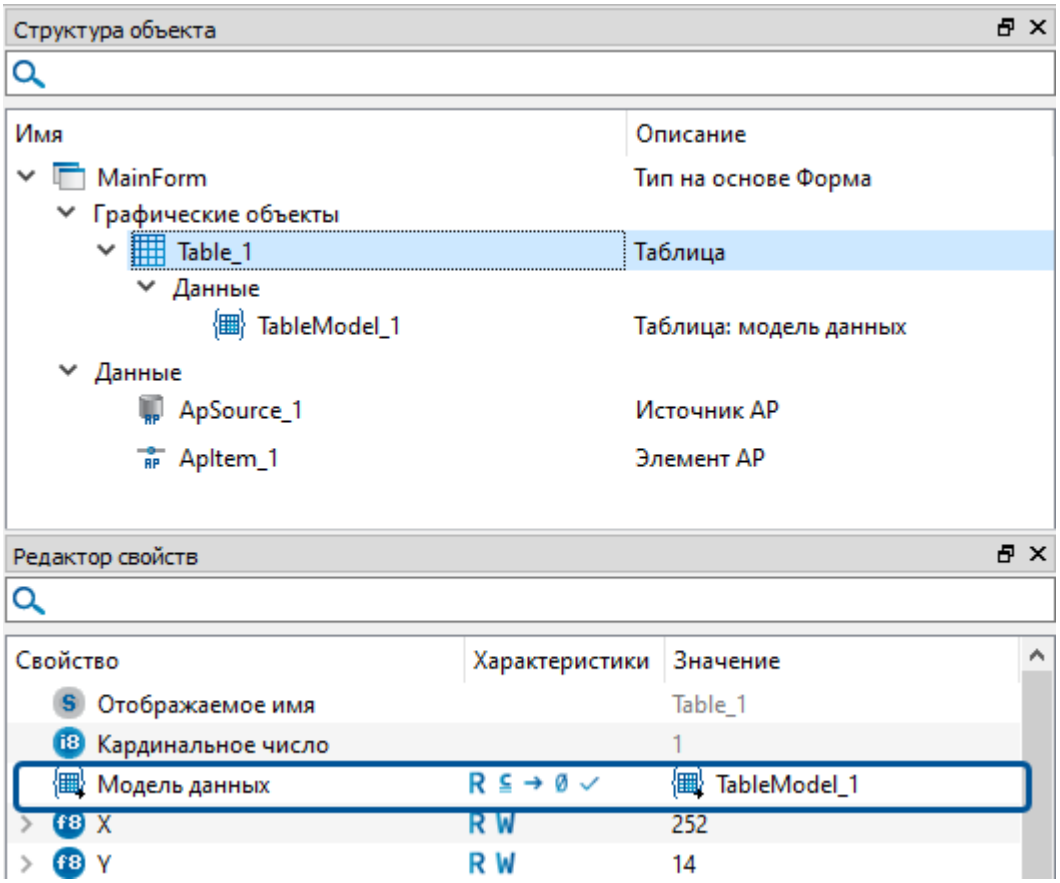
полученные из источника, в таблицу.



2. Добавьте компонент **Таблица: модель данных** дочерним компоненту **Таблица**.



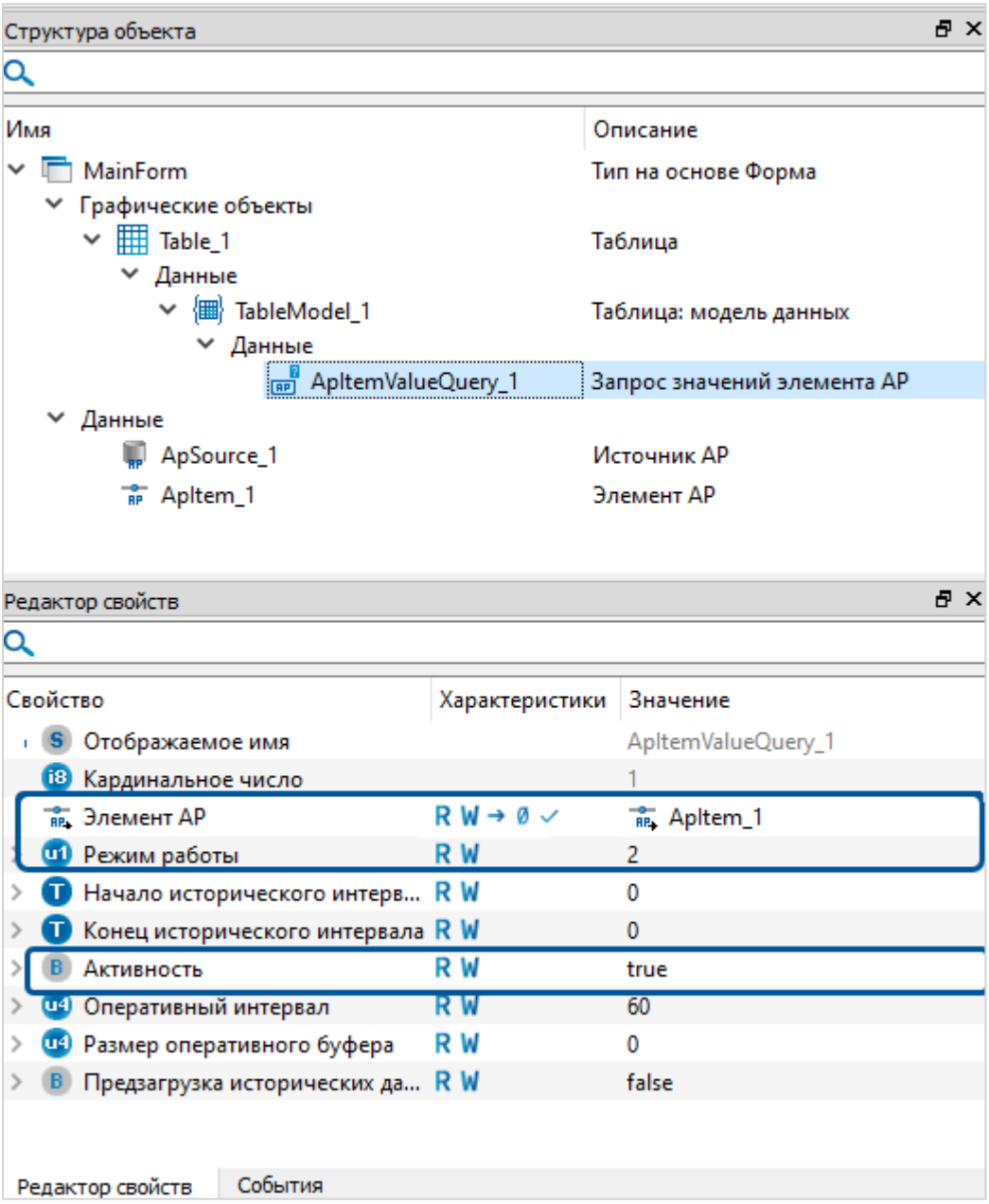
3. В редакторе свойств у объекта **Таблица** укажите ссылку на добавленный объект **Таблица: модель данных**. Для этого нажмите правой кнопкой мыши по свойству Модель данных → **Сослаться** → **TableModel_1.**



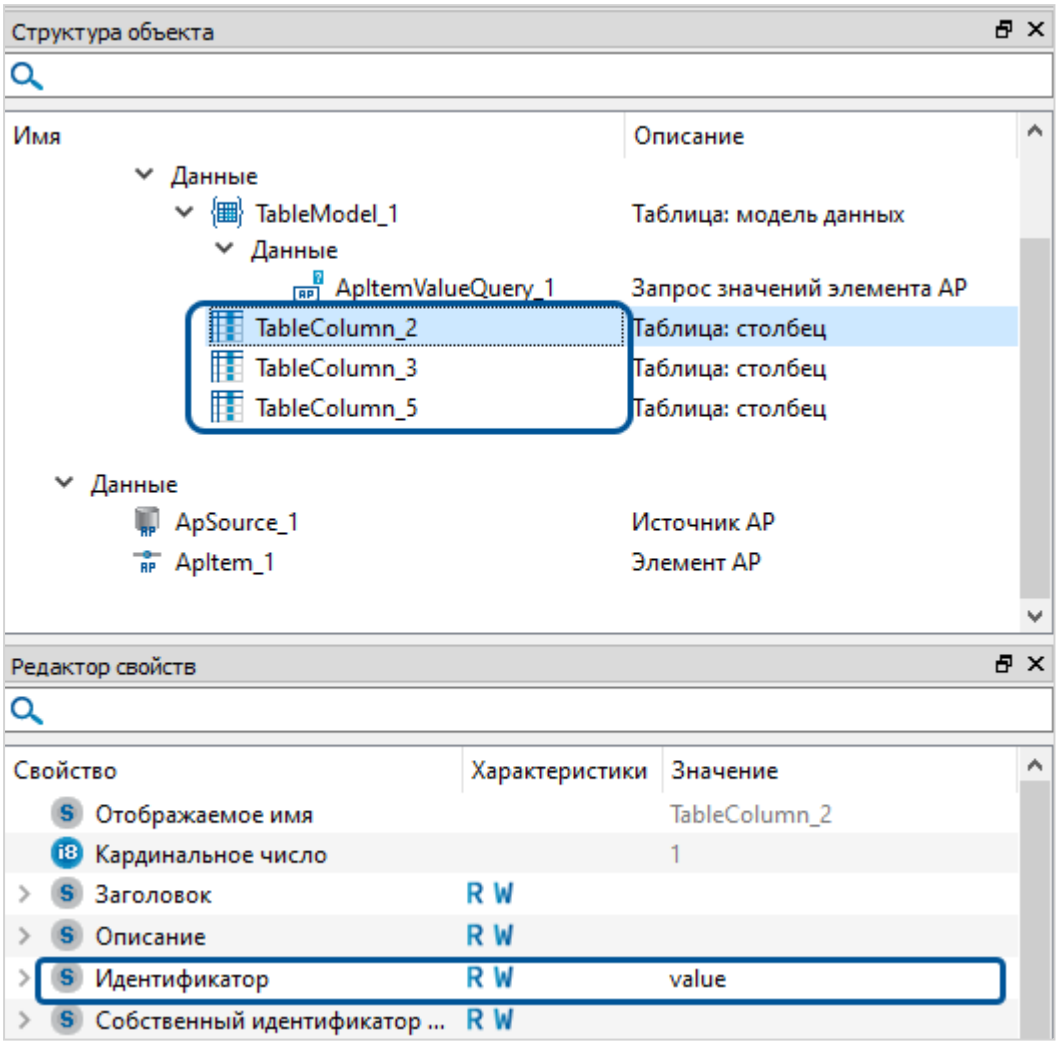
4. Добавьте компонент **Запрос значений элемента АР** дочерним компоненту **Таблица: модель данных**. В свойстве **Элемент АР** укажите ссылку на добавленный ранее объект типа **Элемент АР**. Для этого правой кнопкой мыши нажмите по свойству **Элемент АР**→ **Сослаться** → **ApItem_1**.

4.1. Активируйте компонент в свойстве **Активность** указав значение **«True»**.

4.2. В свойстве **Режим работы** выберите **«2»** для получения оперативных значений:

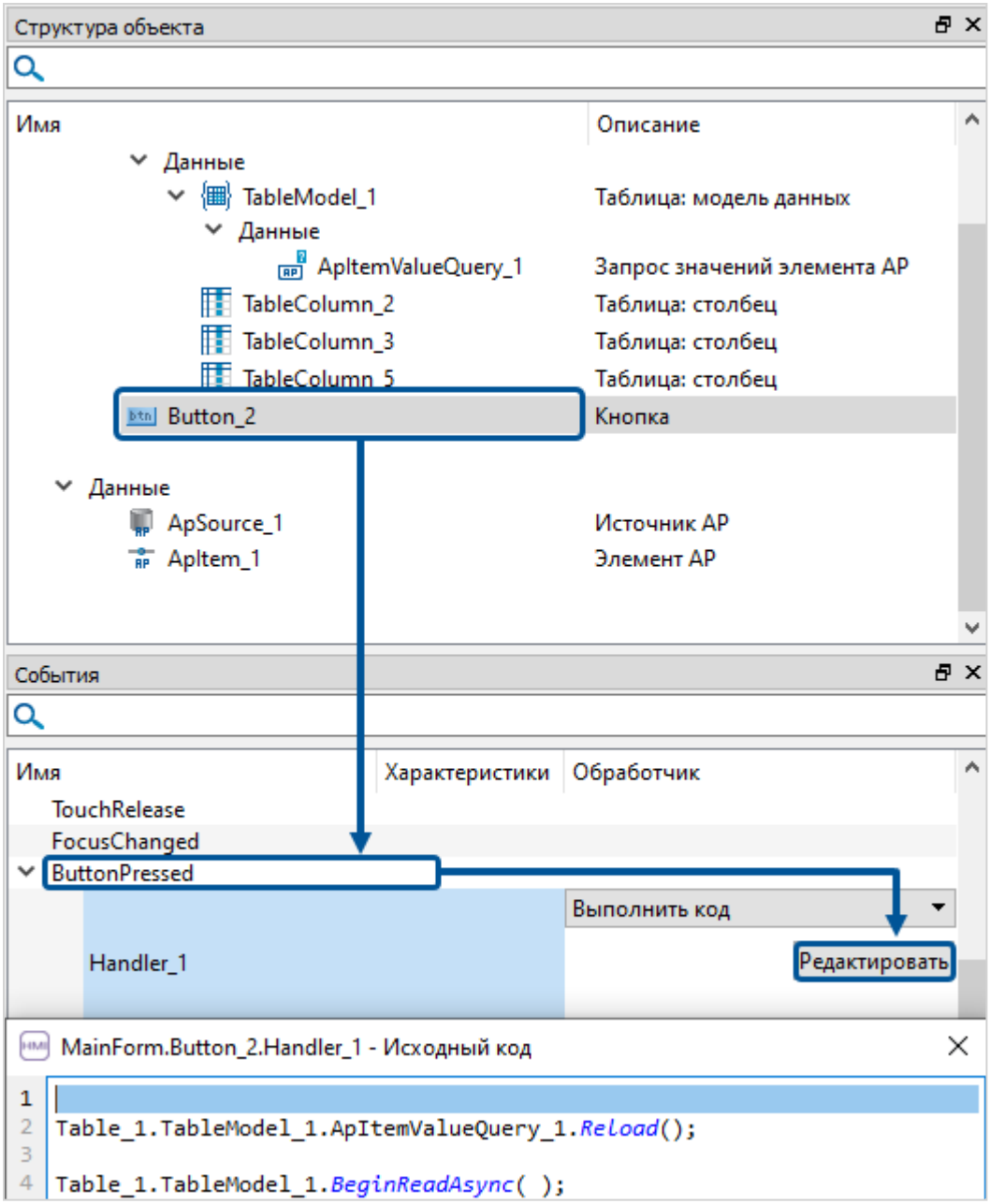


5. Добавьте компоненту **Таблица** дочерние компоненты **Таблица: столбец** с идентификаторами:
   - **«value»**
   - **«quality»**
   - **«timestamp»**



15.2.7.3. Добавить кнопку запроса данных с источника

Добавьте на форму компонент **Кнопка**, по нажатию которой будет происходить запрос данных с источника данных, поставляющего оперативные или исторические значения. В обработчике ButtonPressed пропишите следующий код:



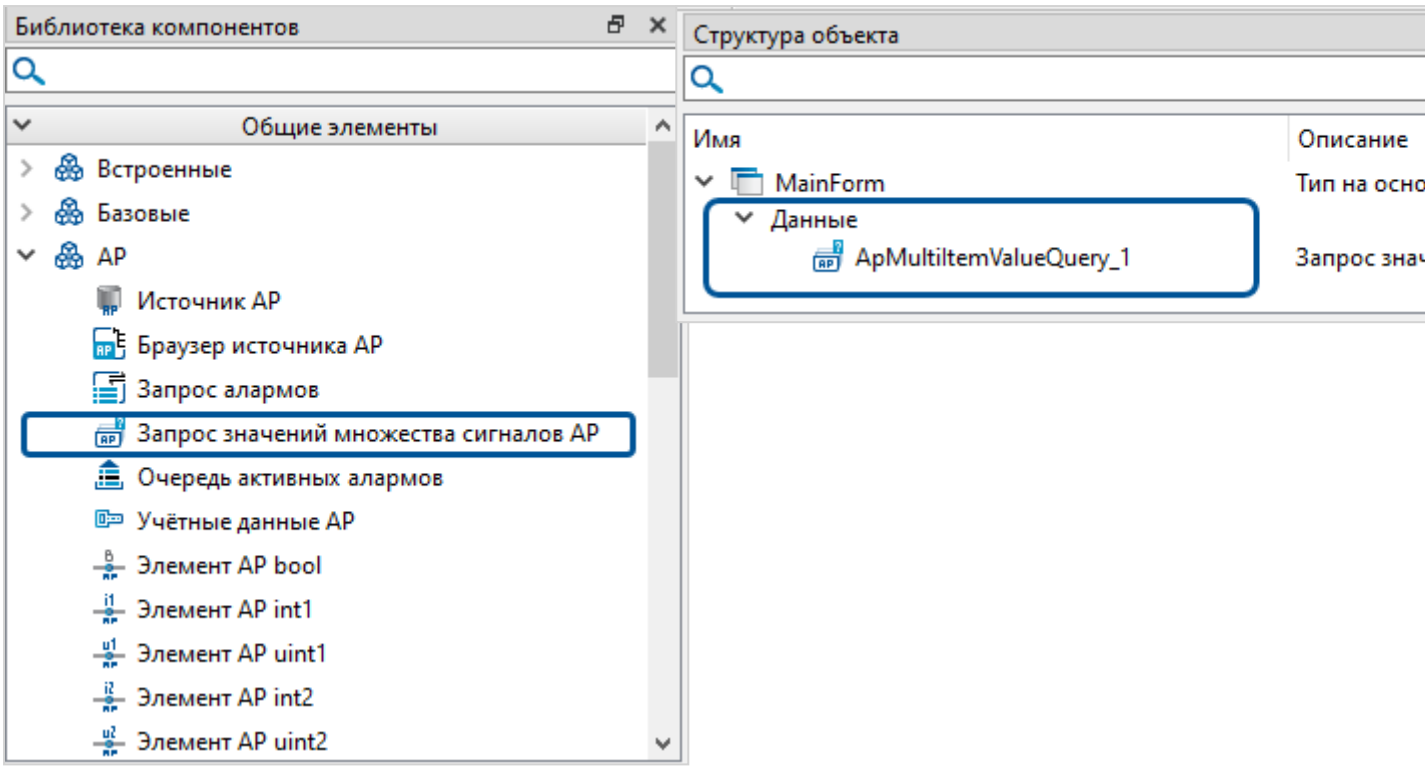

```
Table_1.TableModel_1.ApItemValueQuery_1.Reload(); //запрашиваем данные с источника
Table_1.TableModel_1.BeginReadAsync( ); //заполняем таблицу полученными данными
```

15.2.8. Получение значений множества сигналов

Чтобы получить значения нескольких сигналов с источника данных (например, с SePlatform.Data Server), используйте компонент **Запрос значений множества сигналов АР**. На текущий момент компонент **Запрос значений множества сигналов АР** работает только с историческими значениями.

С помощью компонента **Запрос значений множества сигналов АР** можно получить значение, качество сигнала и метку времени. Полученные данные можно отобразить в визуальной таблице на форме с использованием компонентов **Таблица: модель данных** и **Таблица** из юнита **Таблицы**.

Компонент **Запрос значений множества сигналов АР** расположен в юните **АР Библиотеки компонентов** Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



Свойства и методы компонента **Запрос значений множества сигналов AP** рассмотрены в справочном руководстве.

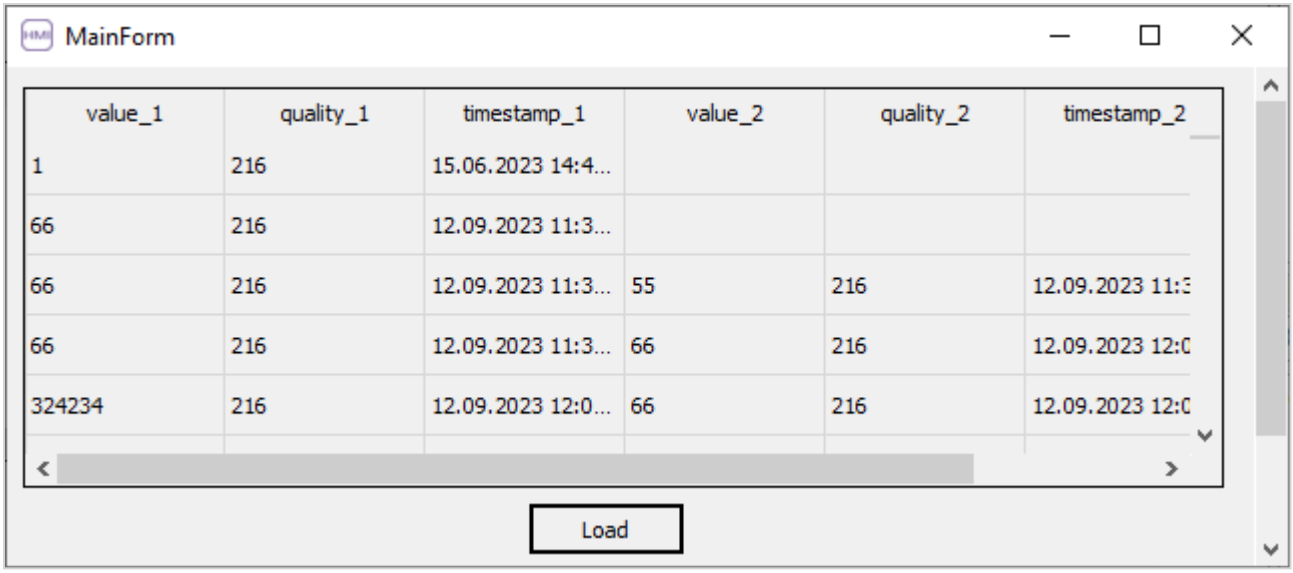
15.2.9. Как настроить запрос значений нескольких сигналов (Демо-проект)

Далее кратко описан небольшой пример (файлы проекта идут в комплекте с документацией)использования компонента **Запрос значений множества сигналов AP** в проекте SePlatform.HMI (проект сделан на языке SePlatform.Om).

В примере показано, как:

- Создать идентификаторы для сигналов **«Deviation.s_deviation4»** и **«Level.s_level4»** и подготовить таблицу для вывода исторических данных.
- Указать перечень сигналов (**«Deviation.s_deviation4»** и **«Level.s_level4»**), с которых требуется запросить данные.
- Указать временной интервал, в пределах которого требуется запросить исторические значения.
- Добавить кнопку запроса данных с источника, поставляющего исторические значения.

В результате будет подготовлена форма, на которой при нажатии на кнопку **Load** исторические значения с двух сигналов будут получены с SePlatform.Data Server и помещены в таблицу.

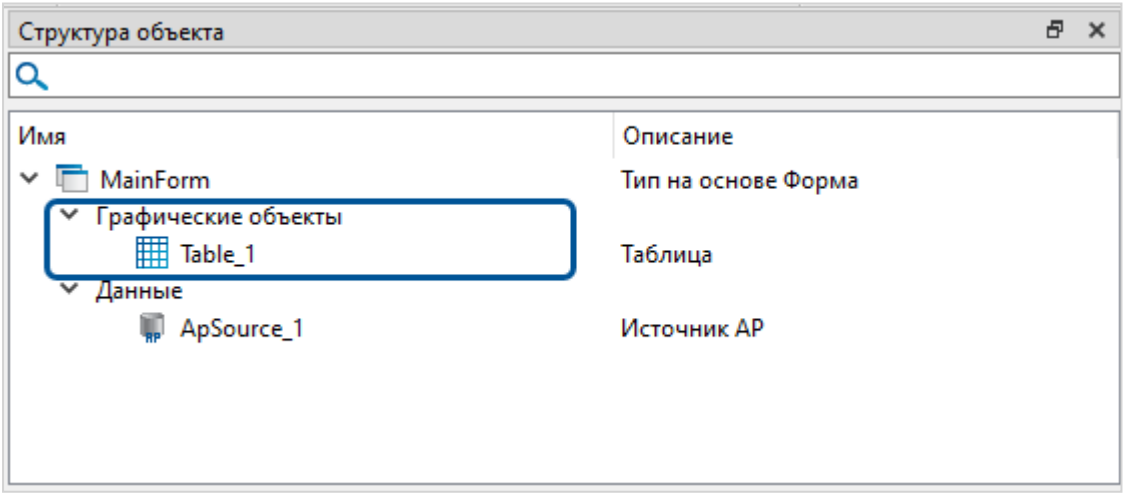


ПРИМЕЧАНИЕ

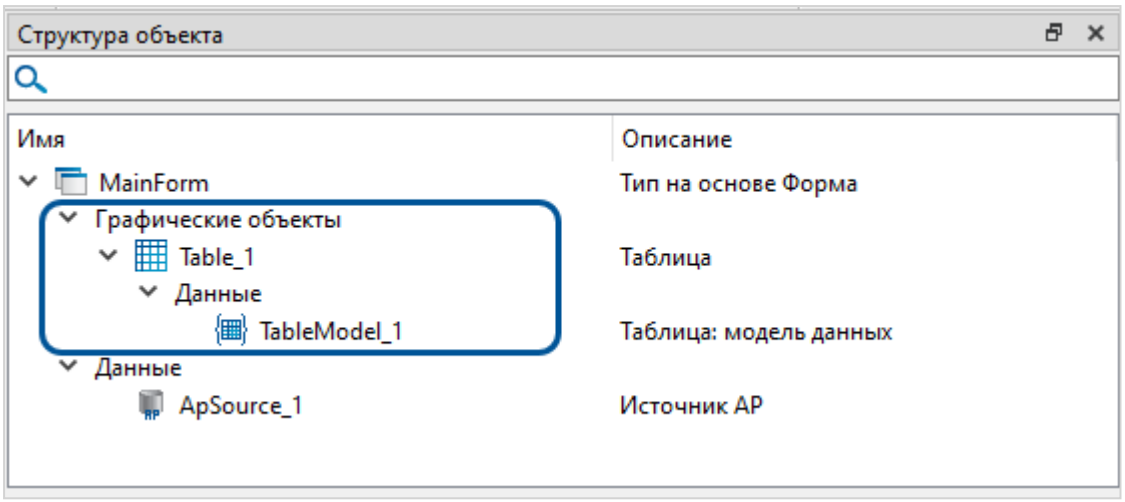
Убедитесь, что у вас уже настроено подключение к источнику данных, с которым вы планируете взаимодействовать. Для получения подробной информации о настройке и подключении источника данных обратитесь к разделу Подключение к источнику (стр. 147).

15.2.9.1. Создать идентификаторы для сигналов и подготовить таблицу для вывода исторических данных

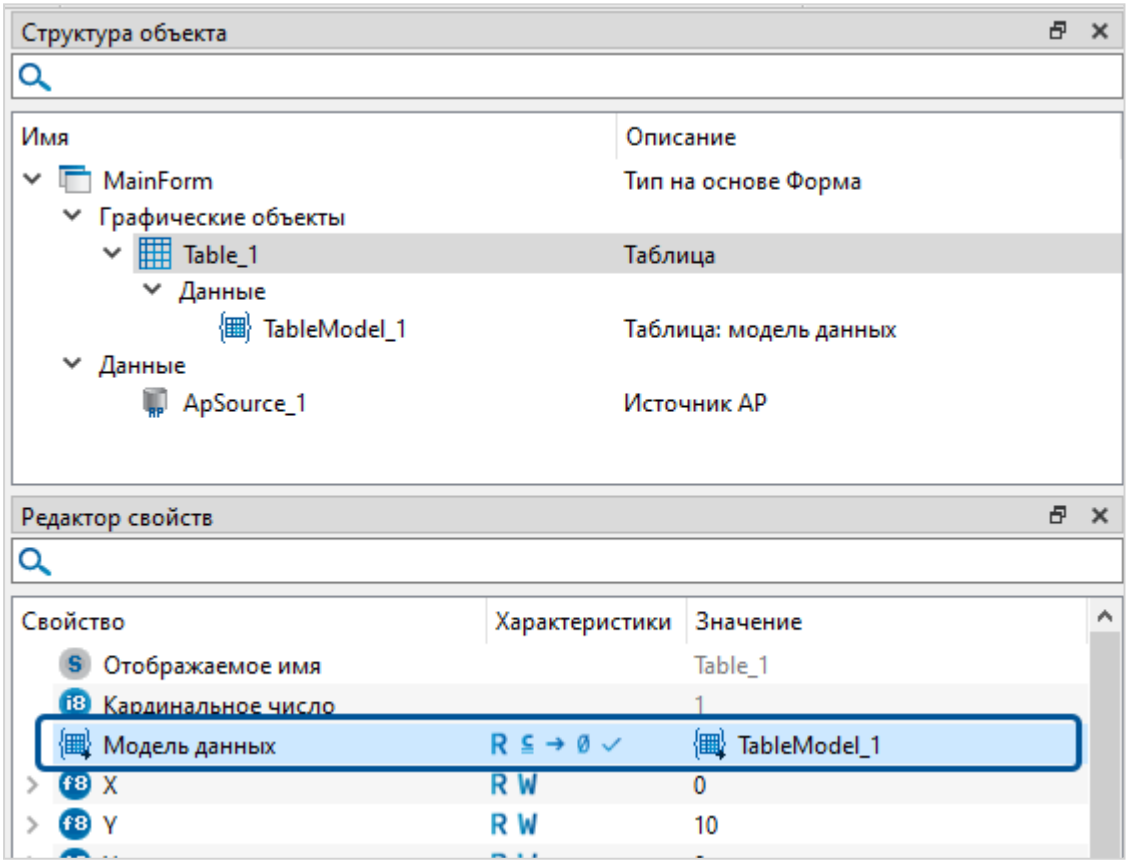
1. Добавьте на экранную форму объект типа **Таблица**, чтобы иметь возможность помещать данные, полученные из источника, в таблицу.



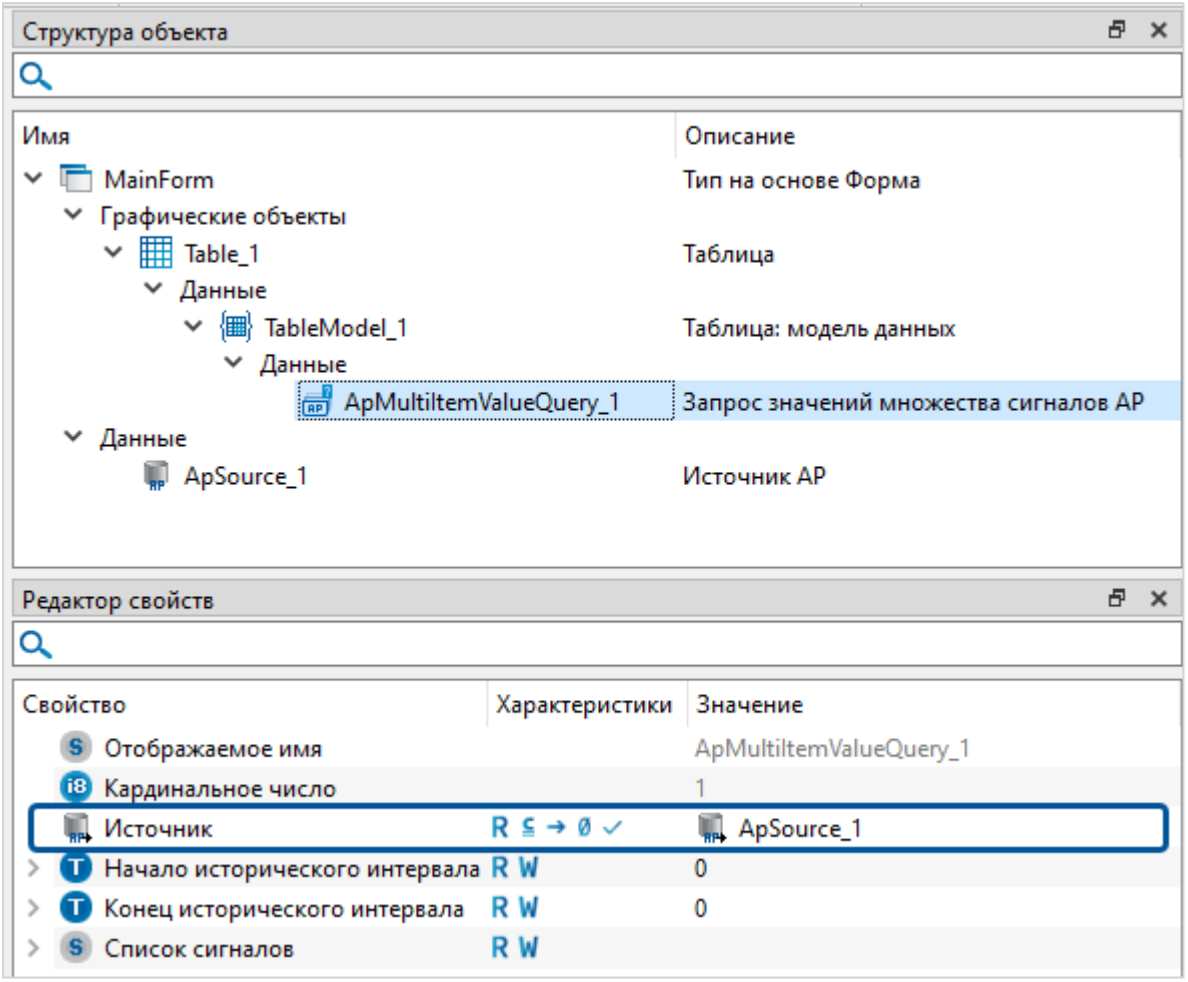
2. Добавьте компонент **Таблица: модель данных** дочерним компоненту **Таблица**.



3. В редакторе свойств у объекта **Таблица** укажите ссылку на добавленный объект **Таблица: модель данных**. Для этого нажмите правой кнопкой мыши по свойству **Модель данных** →**Сослаться**→ **TableModel_1**.



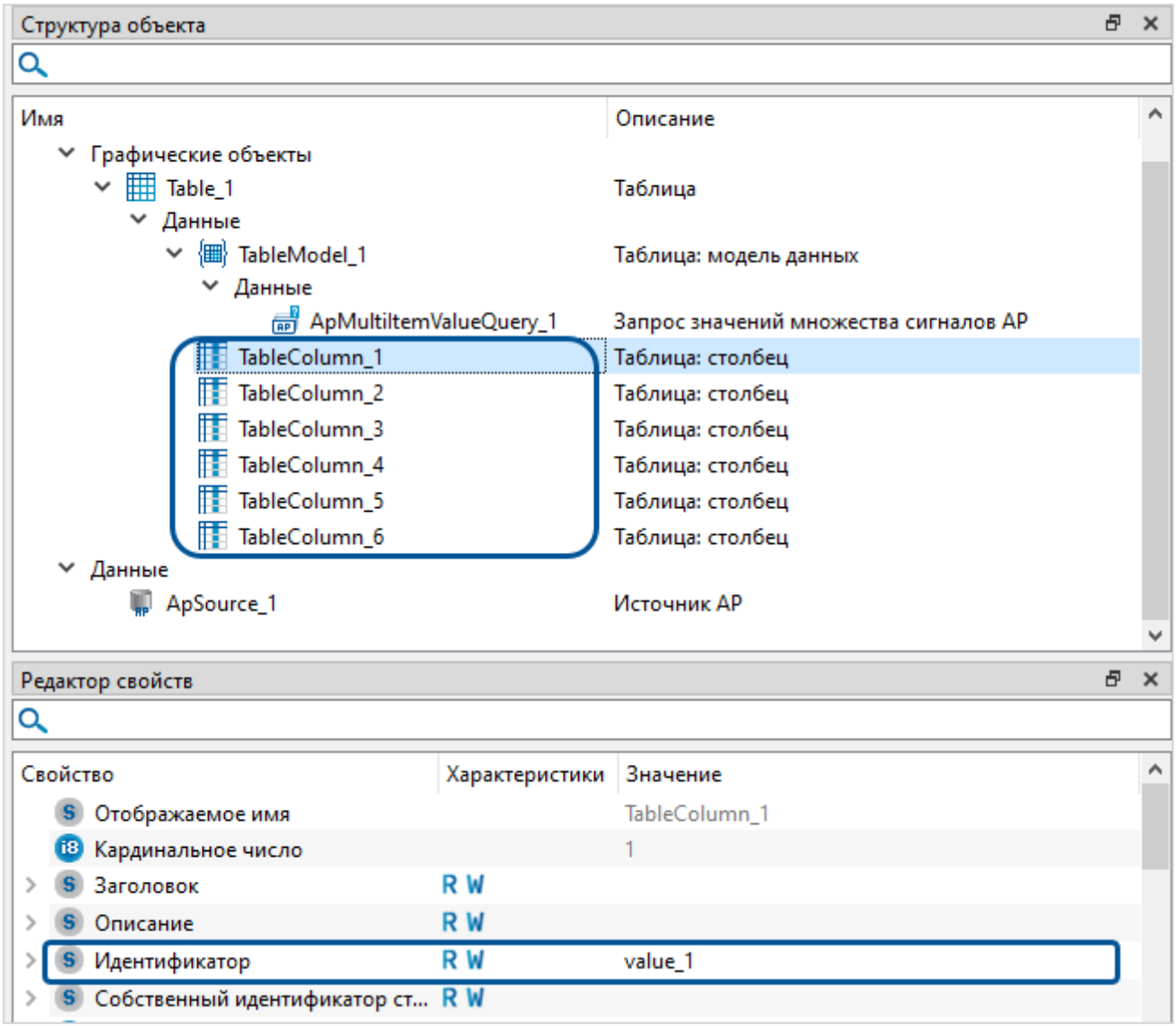
4. Добавьте компонент **Запрос значений множества сигналов AP** дочерним компоненту **Таблица: модель данных**. В свойстве Источник укажите ссылку на добавленный ранее объект типа **Источник AP**. Для этого правой кнопкой мыши нажмите по свойству **Источник** → **Сослаться** → **ApSource_1**.



5. Добавьте компоненту **Таблица** дочерние компоненты **Таблица: столбец** для данных первого сигнала **«Deviation.s_deviation4»** с идентификаторами:
   - **«value_1»**
   - **«quality_1»**
   - **«timestamp_1»**

   Аналогично, для данных второго сигнала **«Level.s_level4»** добавьте столбцы с идентификаторами:
   - **«value_2»**
   - **«quality_2»**
   - **«timestamp_2»**

Идентификаторы могут быть введены в свободной форме. На следующем шаге при заполнении свойства Список сигналов используйте выбранные идентификаторы при передаче JSON-структуры.



15.2.9.2. Указать список сигналов, с которых будут запрошены данные

Чтобы указать перечень сигналов (например, **«Deviation.s_deviation4»** и **«Level.s_level4»**), с которых будут запрошены данные , настройте свойство Список сигналов для компонента **Запрос значений множества сигналов AP**. В свойстве введите JSON-структуру с указанием тегов сигналов на сервере, а также соответствующие идентификаторы (те идентификаторы, которые были введены на предыдущем этапе) столбцов, в которые будут записываться значения, качество и метки времени сигналов:

```
{
 "items": [
 {
    "tag": "Deviation.s_deviation4",
    "value": "value_1",
    "quality":"quality_1",
    "timestamp":"timestamp_1"
 },
 {
    "tag": "Level.s_level4",
    "value": "value_2",
    "quality":"quality_2",
    "timestamp":"timestamp_2"
 }
           ]
}
```

15.2.9.3. Указать временной интервал, в пределах которого требуется запросить исторические данные

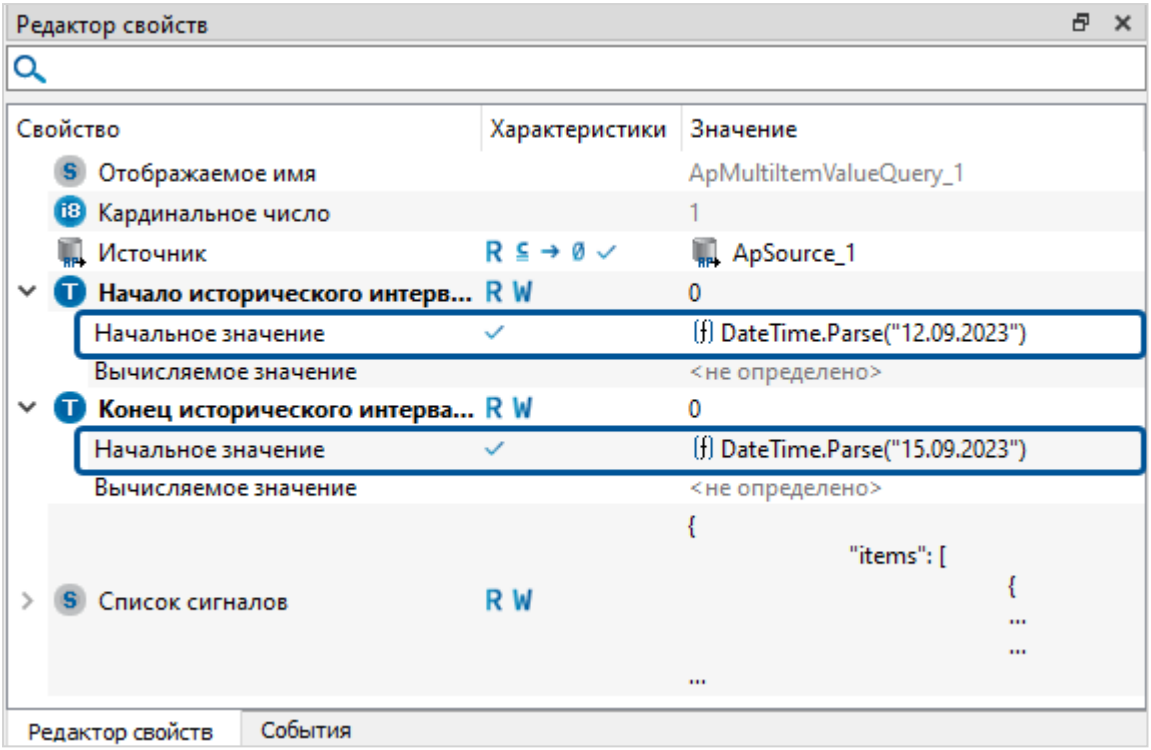
Для указания временного интервала запроса исторических данных настройте свойства **Начало исторического интервала** и **Конец исторического интервала**. Правой кнопкой мыши нажмите на **Начальное значение**, в контекстном меню выберите **Задать формулу**.

Для указания начального значения **«12.09.2023»** введите формулу:

```
DateTime.Parse("12.09.2023").
```

Для указания конечного значения **«15.09.2023»** введите формулу:

```
DateTime.Parse("15.09.2023").
```



15.2.9.4. Добавить кнопку запроса данных с источника, поставляющего исторические значения.

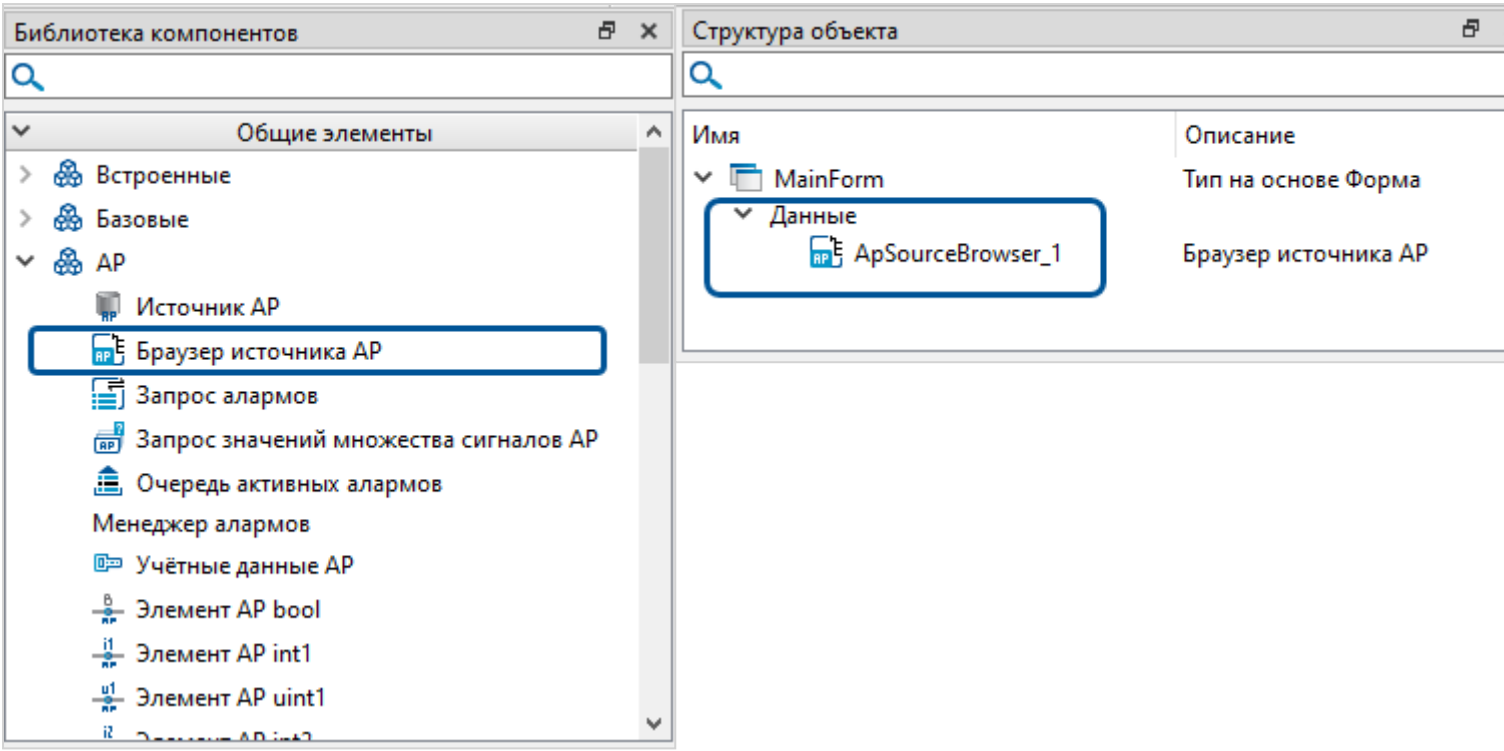
Добавьте на форму компонент **Кнопка**, по нажатию которой будет происходить запрос данных с источника данных, поставляющего исторические значения. В обработчике ButtonPressed пропишите следующий код:

```
Table_1.TableModel_1.ApMultiItemValueQuery_1.Reload(); //запрашиваем данные с источника
Table_1.TableModel_1.BeginReadAsync( ); //заполняем таблицу полученными данными
```

15.2.10. Получение информации о структуре узлов источника данных

Чтобы просматривать дерево сигналов источника данных (например, SePlatform.Data Server), а также осуществлять поиск узлов в этом источнике по заданному фильтру воспользуйтесь компонентом **Браузер источника АР**.

Компонент **Браузер источника АР** расположен в юните **АР Библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



С помощью компонента можно получить информацию о структуре источника данных, включая имена узлов, их идентификаторы, описания, а также наличие дочерних элементов.

Полученные данные можно отобразить в виде древовидной структуры на форме с использованием компонентов **Источник данных дерева**, **Дерево** и **Колонка дерева** из юнита **«Визуальные»**. Чтобы узнать подробности о возвращаемых значениях, обратитесь к разделу Какие данные возвращает компонент.

Подробнее о свойствах и методах компонента **Браузер источника AP**.

15.2.10.1. Какие данные возвращает компонент

Компонент **Браузер источника AP** возвращает данные в табличном формате. Используя указанные ниже идентификаторы, вы можете обращаться к этим данным в проекте. Для просмотра результатов браузинга подключите **Браузер источника AP** к одному из компонентов: **Дерево** или **Таблица**.

Идентификатор колонки	Тип колонки	Тип возвращаемого значения	Описание
name	1	string	Имя узла.
id	2	string	Идентификатор элемента.
parent_id	3	string	Идентификатор родительского узла.
display_name	4	string	Отображаемое имя.

cdt	5	uint1	Идентификатор типа значения, хранимого в узле источника. › **«0»** - empty; › **«1»** - Int1; › **«3»** - UInt1; › **«5»** - Bool; › **«6 »**- UInt4; › **«7»** - Int4; › **«8»** - UInt2; › **«9»** - Int2; › **«12»** - UInt8; › **«13»** - Int8; › **«14»** - Float; › **«15»** - Double; › **«17»** - String; › **«18»** - Time.
historizing	5	bool	Флаг ведения истории.
description	5	string	Описание узла.
has_children	6	bool	Наличие у узла дочерних узлов.

15.2.11. Как построить дерево сигналов источника данных SePlatform.Data Server (Демо-проект)

Далее кратко описан небольшой пример (скачать его можно здесь) использования компонента **Браузер источника АР** в проекте SePlatform.HMI (проект сделан на языке SePlatform.Om).

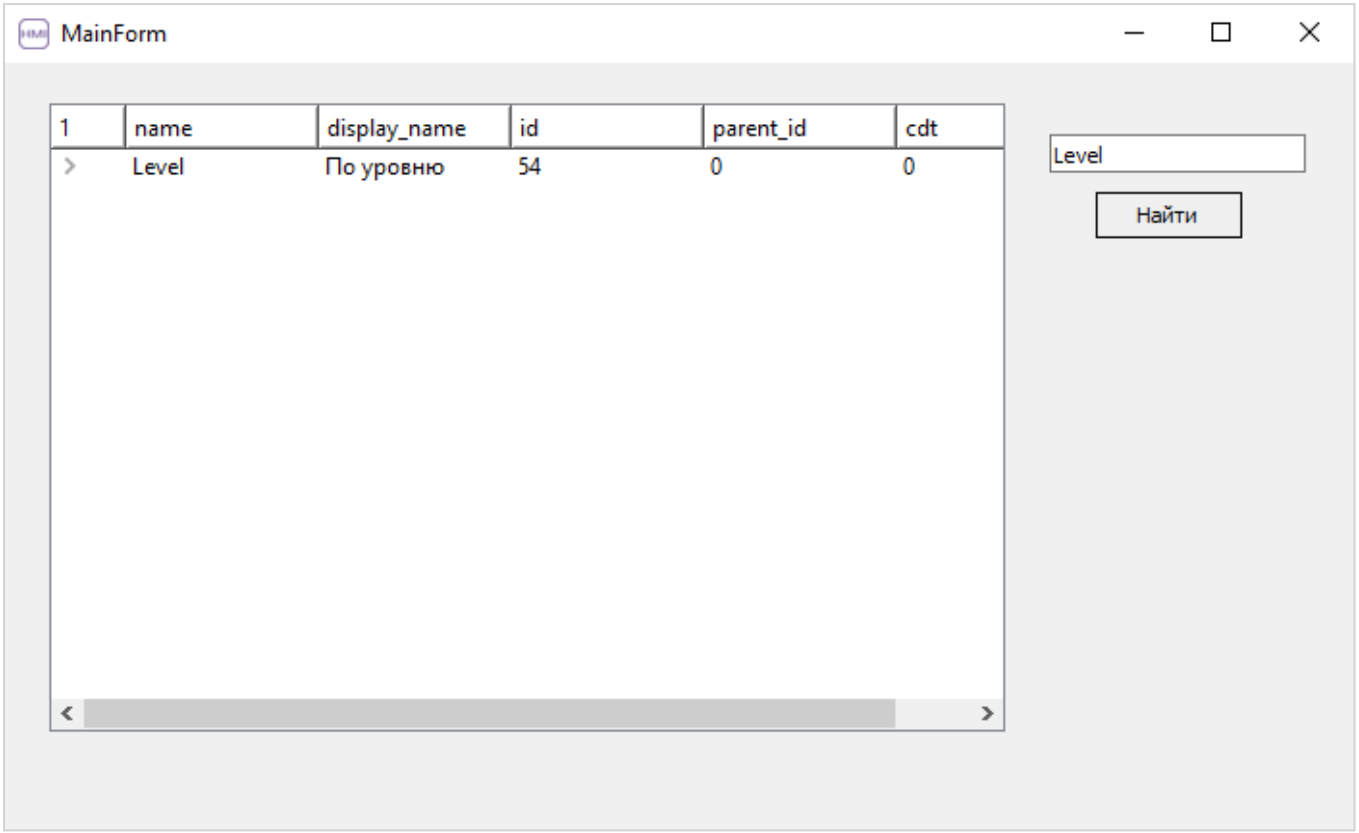
В примере показано, как:

› Настроить отображение дерева сигналов источника данных в виде древовидной структуры;

› Настроить взаимодействие компонента **Браузер источника АР** с источником данных для получения информации о структуре узлов источника;

› Получить информацию об узлах источника, имя которых соответствует заданному фильтру.

В результате будет подготовлена форма, при открытии которой информация о структуре источника будет получена с SePlatform.Data Server и отображена в виде древовидной структуры с возможностью ввода фильтра для поиска узлов по имени.

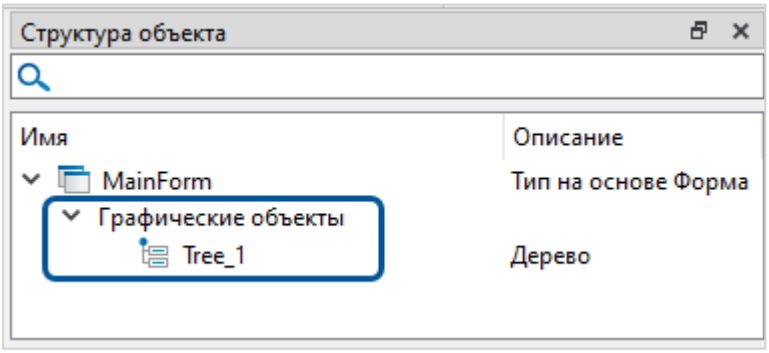


ПРИМЕЧАНИЕ

Убедитесь, что у вас уже настроено подключение к источнику данных, с которым вы планируете взаимодействовать. Для получения подробной информации о настройке и подключении источника данных обратитесь к разделу Подключение к источнику (стр. 147).

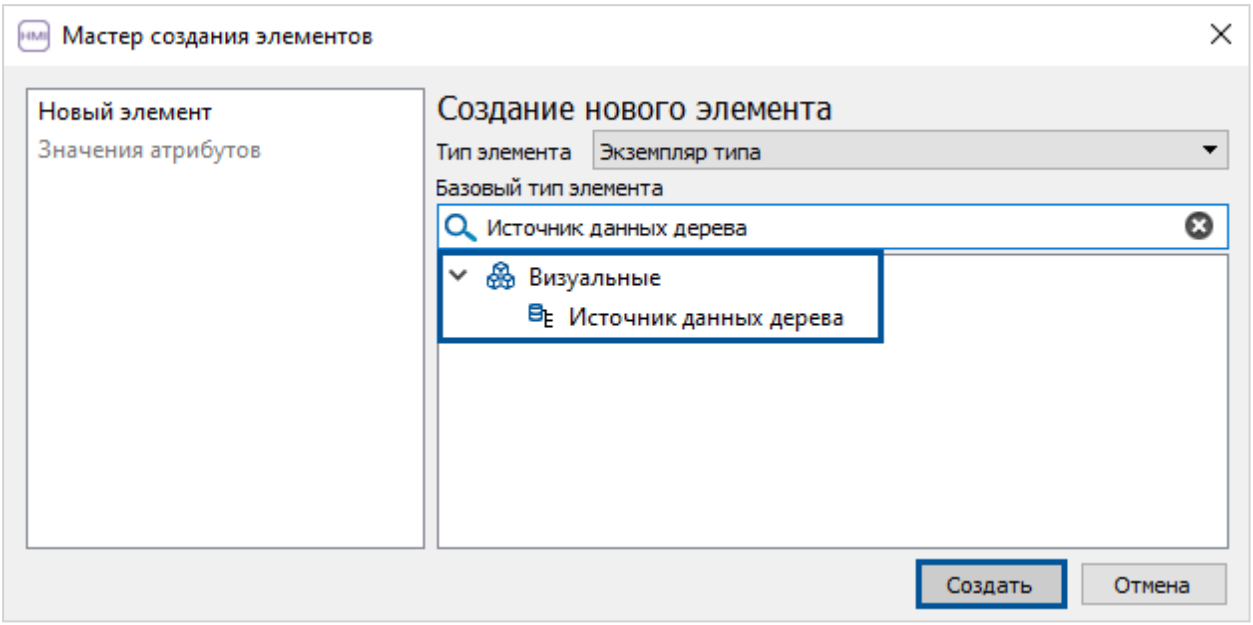
15.2.11.1. Настроить отображение дерева сигналов источника данных в виде древовидной структуры

1. Добавьте на экранную форму объект типа **Дерево**.

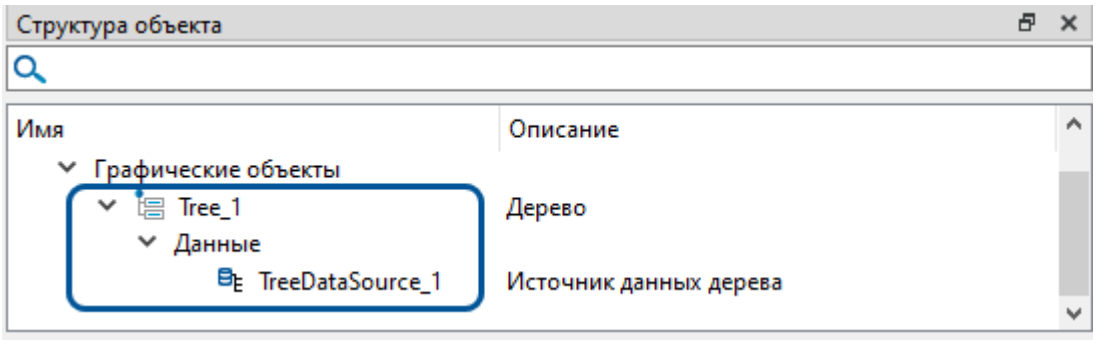


2. В **Структуре объектов** щелкните правой кнопкой мыши по компоненту **Дерево**→ **Создать**. Откроется мастер создания элементов.

2.1. В поле **Тип элемента** оставьте **Экземпляр типа** , **Базовый тип элемента** → **Источник данных дерева** (раздел Визуальные) → **Создать**.



2.2. В **Структуре объектов** добавится компонент **Источник данных дерева** дочерним компоненту **Дерево**.



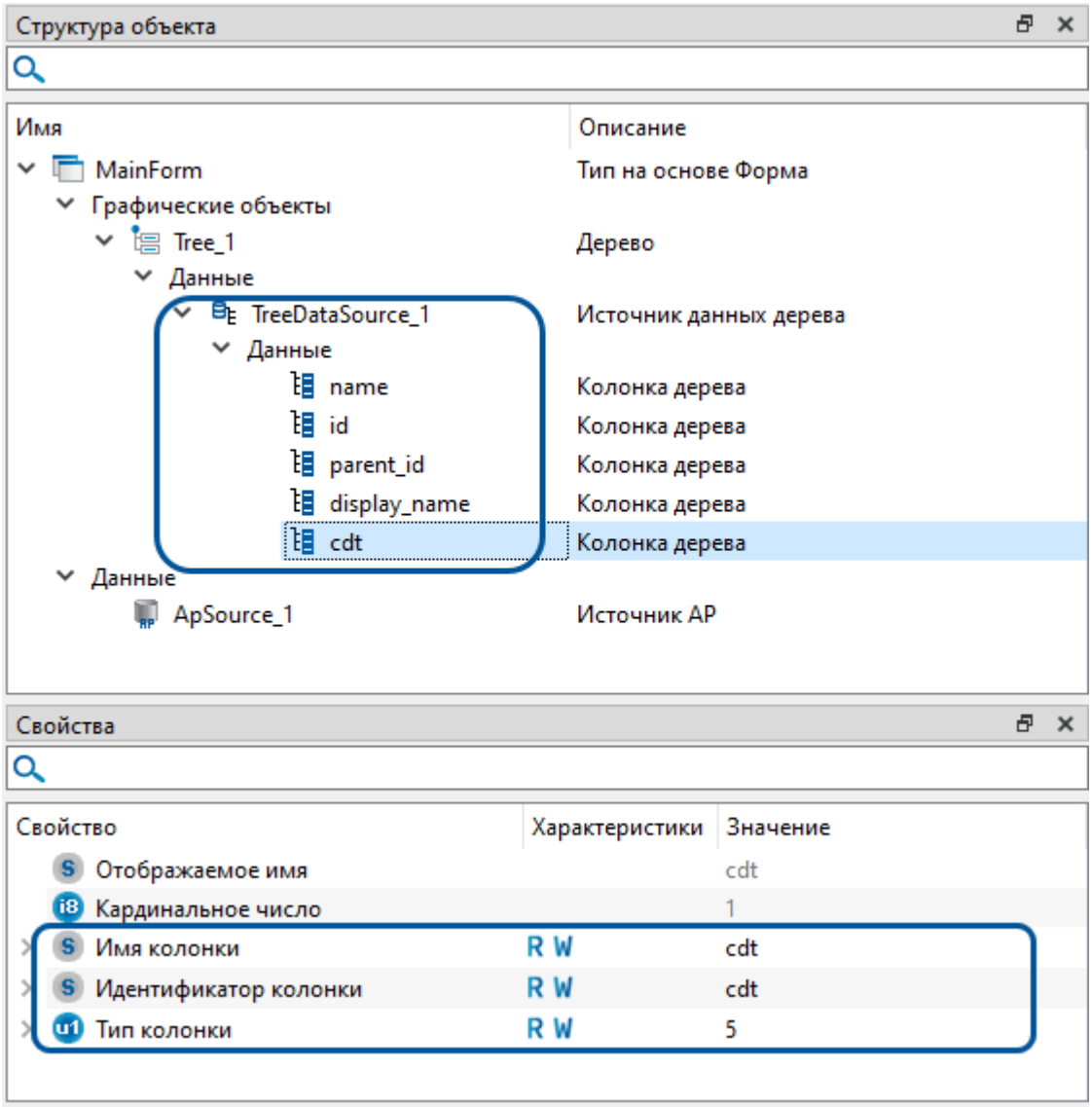
3. Для объекта **Источник данных дерева** аналогичным образом добавьте **Колонки дерева** с идентификаторами:

- **«name»**
- **«id»**
- **«parent_id»**
- **«display_name»**
- **«cdt»**

Используйте значения Идентификатора колонки и Тип колонки в соответствии с таблицей возвращаемых значений в разделе Какие данные возвращает компонент, которая поможет правильно настроить свойства для каждой **Колонки дерева**. Имя колонки указывается в произвольном виде.

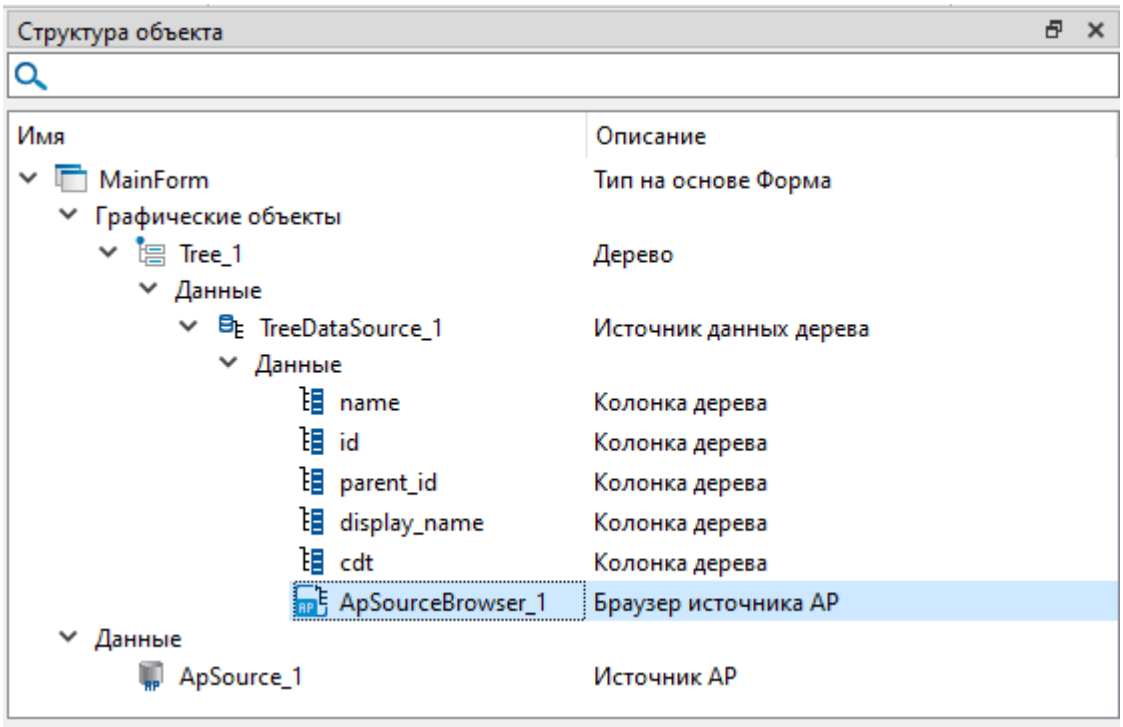
ПРИМЕЧАНИЕ

Колонки дерева с идентификаторами **«name»**, **«id»** и **«parent_id»** обязательны для добавления на форму, все остальные добавляются при необходимости.

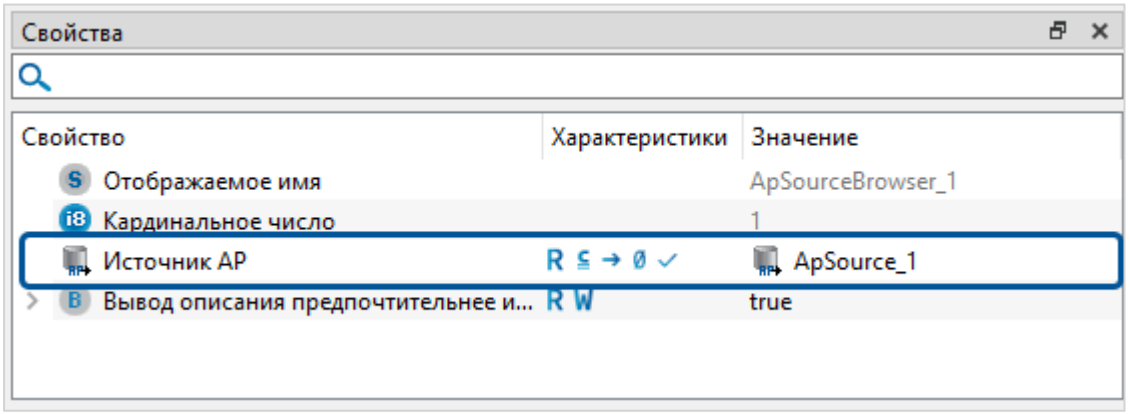


15.2.11.2. Настроить взаимодействие компонента Браузер источника AP с источником данных для получения информации о структуре узлов источника

1. Для просмотра дерева сигналов добавьте на форму компонент **Браузер источника AP** дочерним **Источнику данных дерева**.



2. В редакторе свойств у объекта **Браузер источника AP** укажите ссылку на добавленный ранее компонент **Источник AP**. Для этого нажмите правой кнопкой мыши по свойству Источник AP → **Сослаться** → **ApSource_1**.

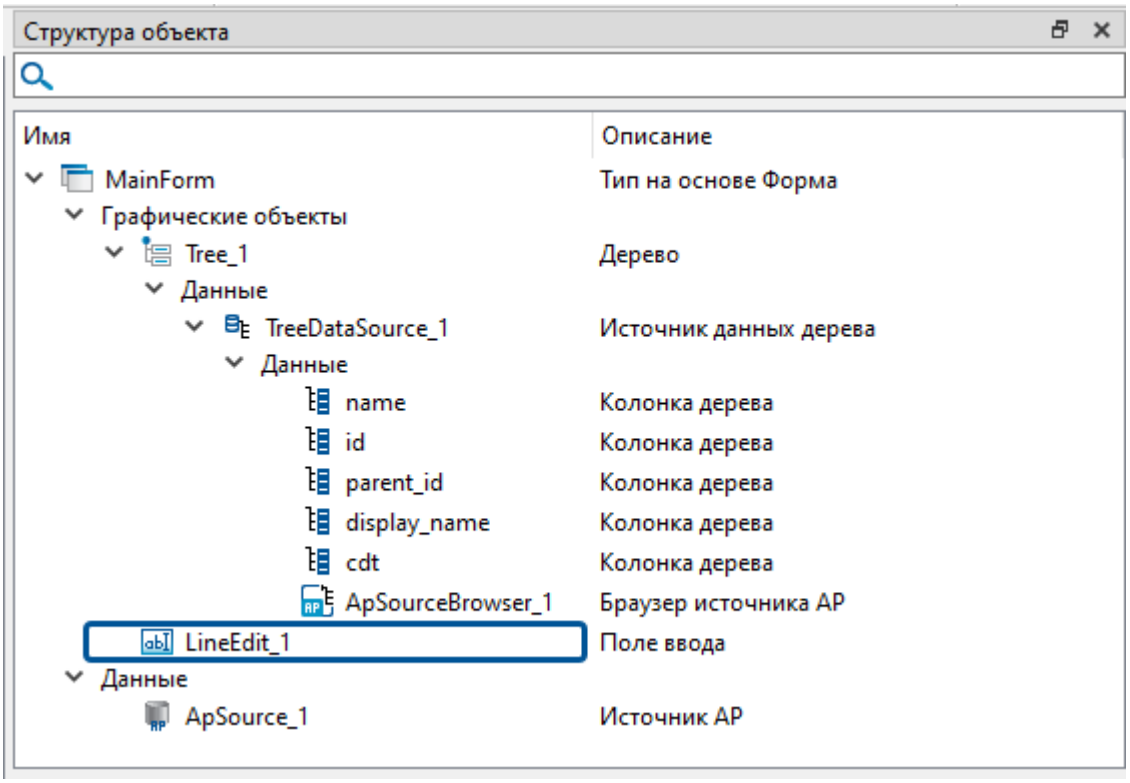


3. Для просмотра дерева сигналов сразу после открытия экранной формы в рантайме, воспользуйтесь обработчиком события Opened. Пропишите в нем следующий код:

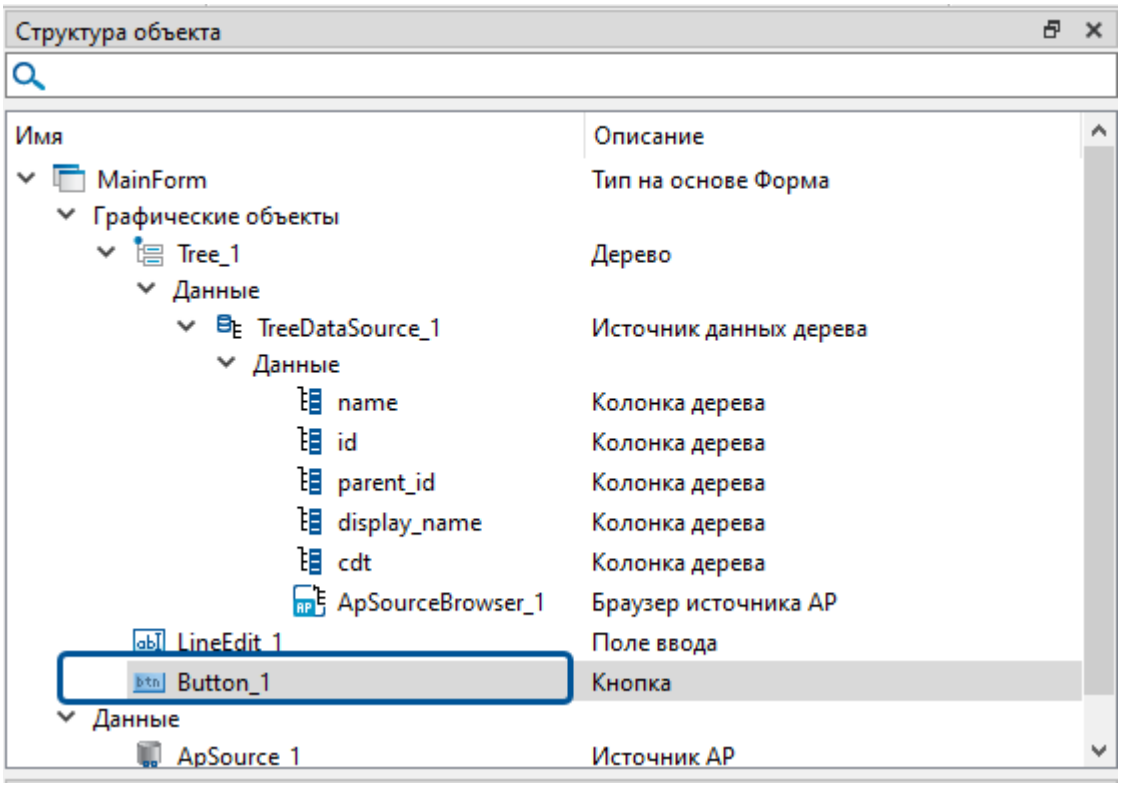
```
Tree_1.TreeDataSource_1.ApSourceBrowser_1.GetChildren(""); //Вызов функции GetChildren
без указания родительского узла, что означает начало браузинга с корня дерева.
```

15.2.11.3. Получить информацию об узлах источника, имя которых соответствует заданному фильтру

1. Добавьте на форму текстовое поле, в котором можно будет вводить фильтр для поиска узлов.



2. Разместите на форме кнопку, которая будет запускать процесс поиска узлов при нажатии на нее.



3. В обработчике события нажатия кнопки ButtonPressed используйте функцию GetSourceNodes(), передав в нее значение фильтра из текстового поля в качестве аргумента. Пример кода:

```
// Получение значения фильтра из текстового поля
filter: string = LineEdit_1.Text;
// Очистка дерева перед выполнением запроса
Tree_1.RemoveChildren("");
// Вызов функции GetSourceNodes() с передачей фильтра
Tree_1.TreeDataSource_1.ApSourceBrowser_1.GetSourceNodes(filter);
```

15.3. Организация иерархии источников данных

ПРИМЕЧАНИЕ

В разделе описана организация иерархии OPC источников данных. Источники AP организуются аналогично.

Чтобы иерархически организовать несколько компонентов **Источник OPC** и **Элемент OPC <T>**, воспользуйтесь свойствами Путь и Родительский источник.

16. Передача данных между объектами

При работе в рантайм из объекта в объект можно перетаскивать данные. Перетаскивать данные можно внутри одного проекта либо между объектами разных проектов.

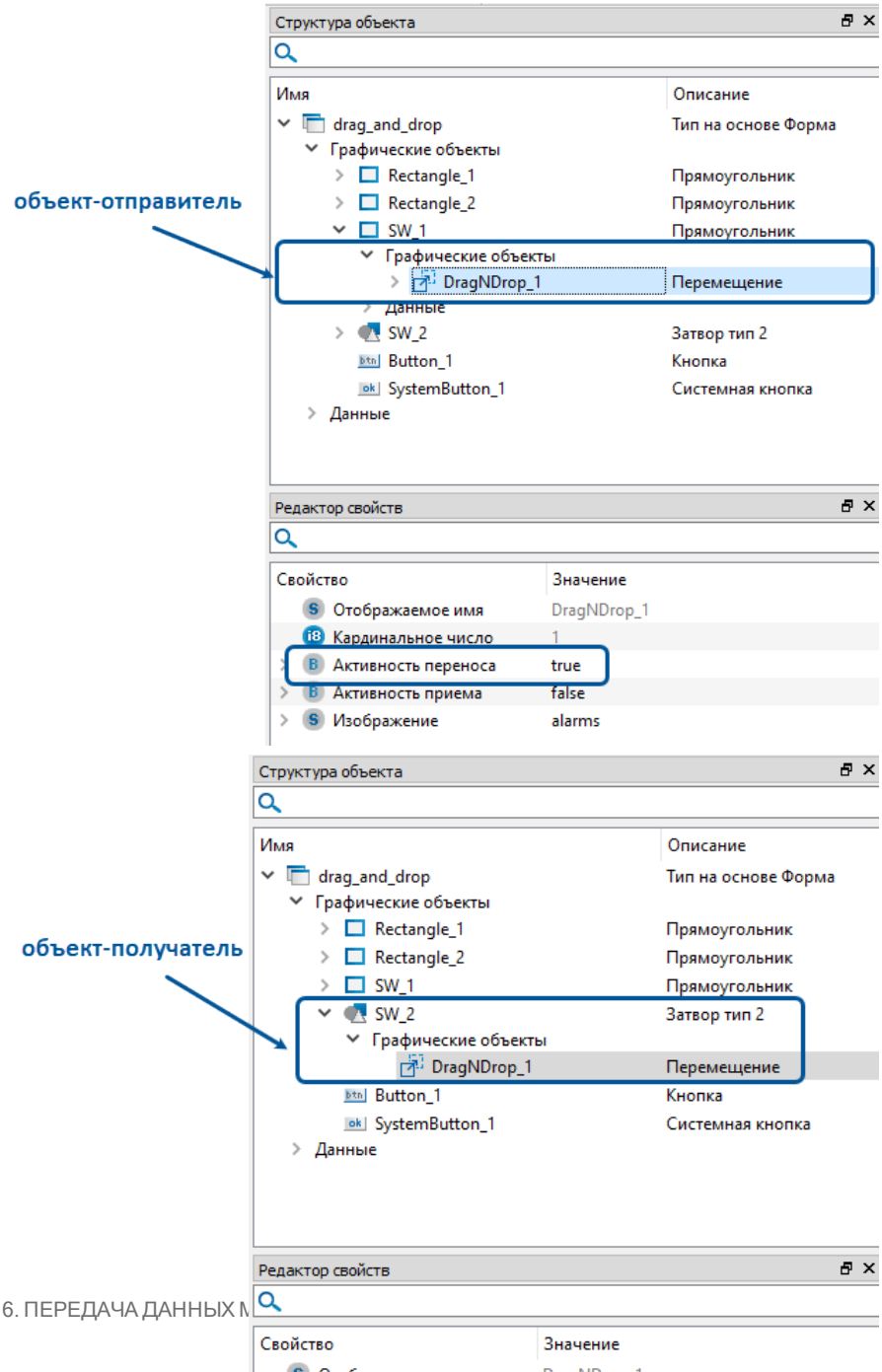
Вы можете передавать между объектами:

- значения свойств объектов/отдельных элементов проекта;
- произвольный текст.

Данные, полученные от другого объекта, можно использовать как угодно, настроив их обработку.

Вы можете перетаскивать одни и те же данные с одного объекта-отправителя на разные объекты-получатели. При этом любой объект может одновременно быть и отправителем и получателем.

Чтобы добавить объекту возможность передавать/принимать данные, добавьте ему дочерний элемент **Перемещение** и разрешите отправку/прием данных с помощью свойств Активность переноса и Активность приема элемента **Перемещение**.



Чтобы указать изображение, отображаемое рядом с курсором при перетаскивании, перейдите к объекту-отправителю и укажите название файла в свойстве **Изображение** элемента **Перемещение**. По умолчанию при переносе отображается изображение объекта-отправителя.
Чтобы создать иллюзию перетаскивания данных, используйте обработчики событий **MousePress** объекта-отправителя и **OnDrop** элемента **Перемещение** на стороне объекта-получателя. Укажите передаваемые данные в обработчике события **MousePress** и настройте обработку принимаемых данных в обработчике события **OnDrop**. Для перетаскивания щелкните на объекте-отправителе и не отпуская клавишу мыши, перетащите данные на объект-получатель.
При перетаскивании на сторону объекта-получателя "сбрасываются" абсолютно все данные, заложенные в обработчик события **MousePress** объекта-отправителя. Какие из "сброшенных" данных будут приняты объектом-получателем, зависит от настройки обработчика события **OnDrop**.
Перед указанием передаваемых данных на стороне объекта-отправителя оцените объем и содержание передаваемых данных и выберите способ передачи.
Способы передачи данных между объектами:

- передача данных единым блоком;
- передача данных частями;
- передача массива бинарных данных.

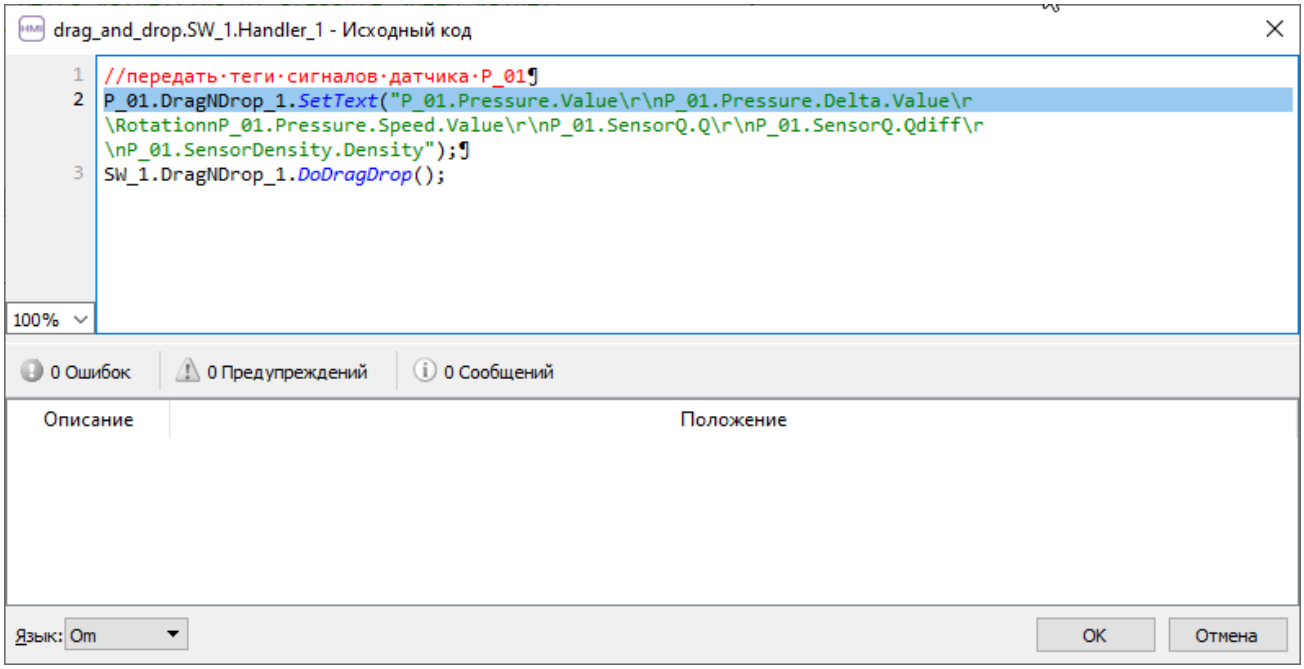
16.1. Передача данных единым блоком

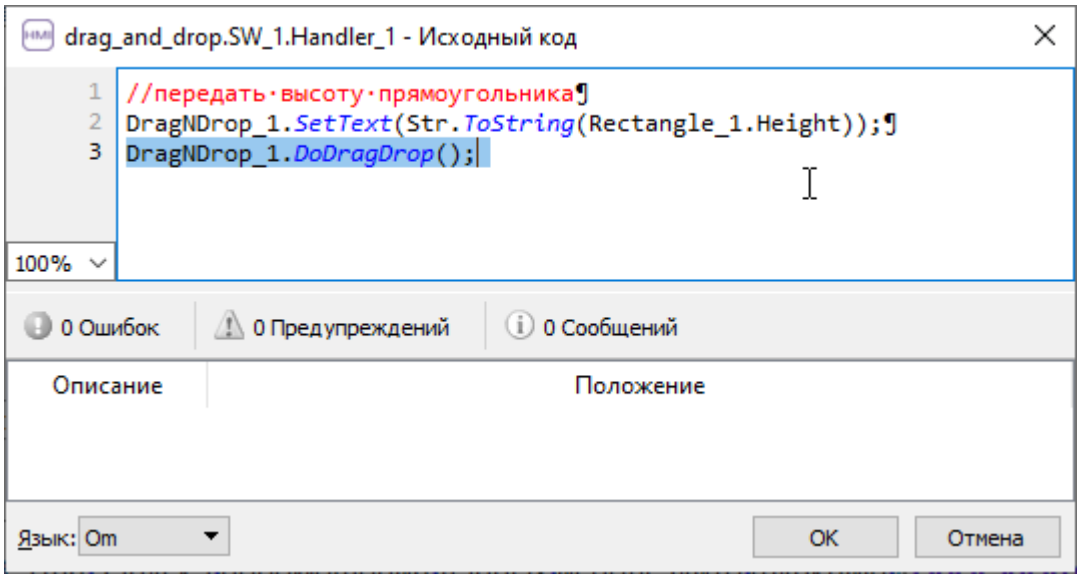
Если вы намерены передать большой блок данных (например, произвольный текст), который целиком нужен каждому объекту-получателю, передавайте данные одним блоком. Для передачи используйте функции **SetText** (**SetHtml**) элемента **Перемещение** на стороне объекта-отправителя и функции **GetText** (**GetHtml**) параметра data события **OnDrop** элемента **Перемещение** на стороне объекта-получателя.

Чтобы настроить передачу единого блока данных, выполните:

1. Для объекта-отправителя и объекта-получателя добавьте дочерний элемент **Перемещение**.

2. В обработчике события MousePress объекта-отправителя укажите передаваемые данные с помощью функции SetText (SetHtml) элемента **Перемещение**.

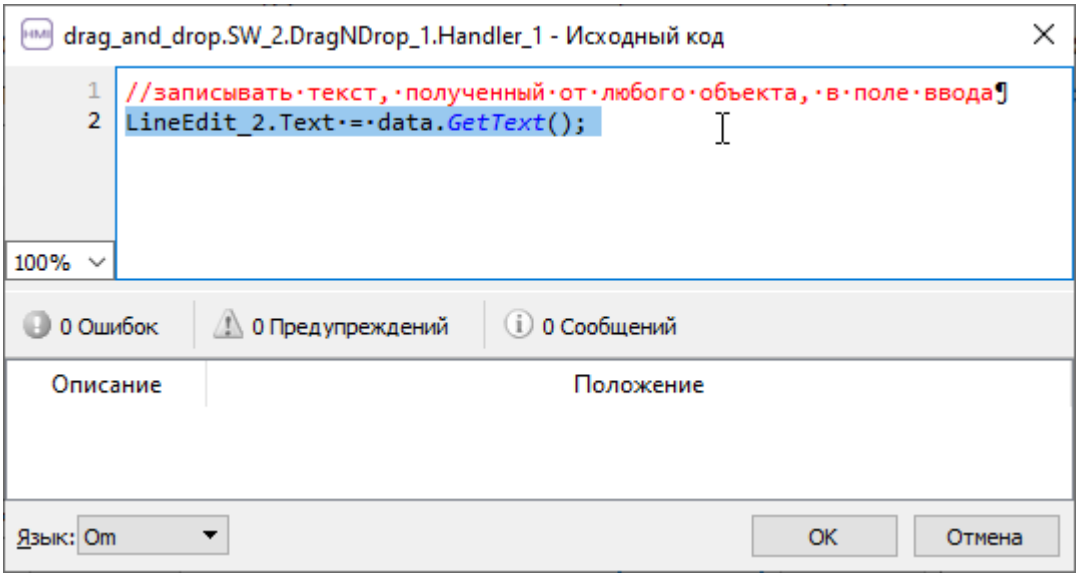




3. Для запуска перемещения данных используйте функцию DoDragDrop элемента **Перемещение**. Вызывайте функцию сразу же после вызова функции SetText (SetHtml) (см. рисунок выше).

4. В обработчике события OnDrop элемента **Перемещение** разрешите объекту-получателю принимать любой текстовый блок с помощью функции GetText (GetHtml) параметра data.

Пропишите как вы будете использовать полученный блок данных. Вы можете по своему усмотрению настроить обработку полученного блока так, чтобы блок после получения разбивался на части. Для этого на стороне объекта-получателя запишите передаваемые данные в виде JSON-структуры или строки с разделителями.



16.2. Передача данных частями

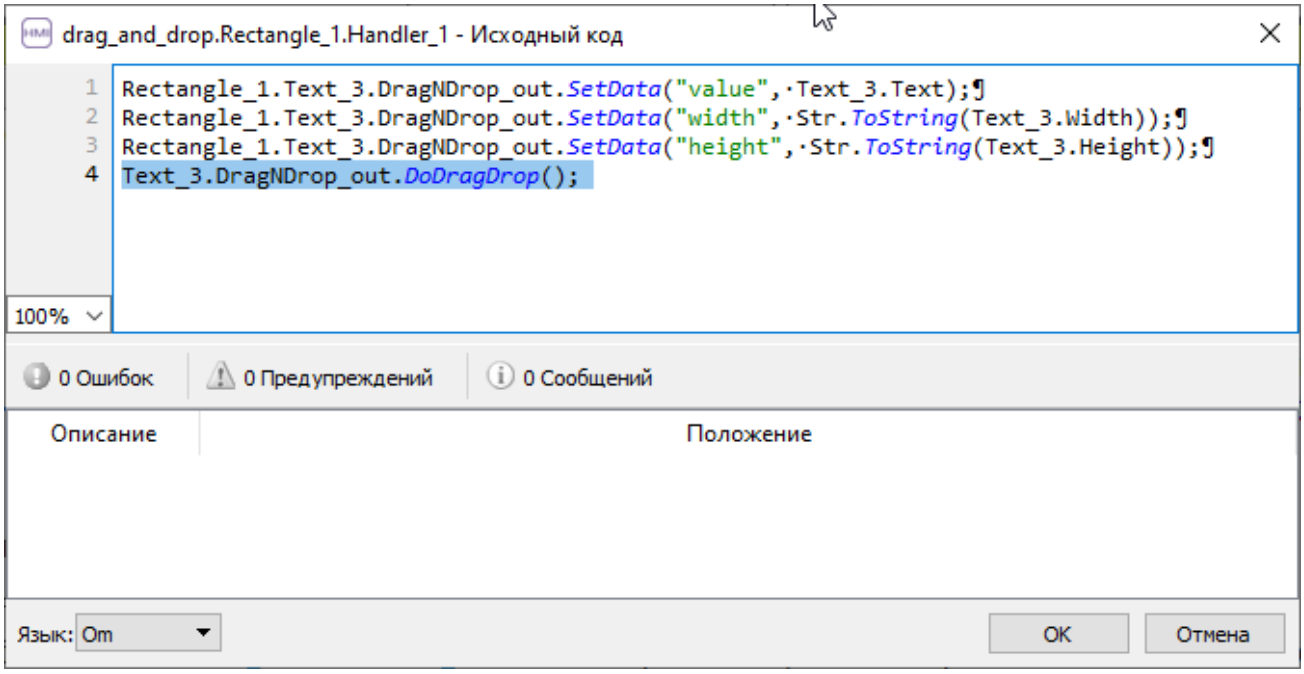
Если каждому из объектов-получателей нужна только определённая часть из всего передаваемого блока, либо вы хотите разбить блок данных на части перед отправкой, используйте функцию SetData элемента **Перемещение** на стороне объекта-отправителя. С помощью функции SetData каждой отдельной части можно присвоить тип (идентификатор). Типы позволяют ограничить объем принимаемых данных на объекте-получателе.

Используйте функцию AddAcceptedFormat элемента **Перемещение**, чтобы указать типы, разрешённые для приёма. При получении блока данных объект-получатель ориентируется на список разрешённых принимаемых типов и принимает только типы из списка. Для приёма типов на стороне объекта-получателя используйте функцию GetData параметра data события OnDrop элемента **Перемещение**.

Чтобы настроить передачу данных по частям, выполните:

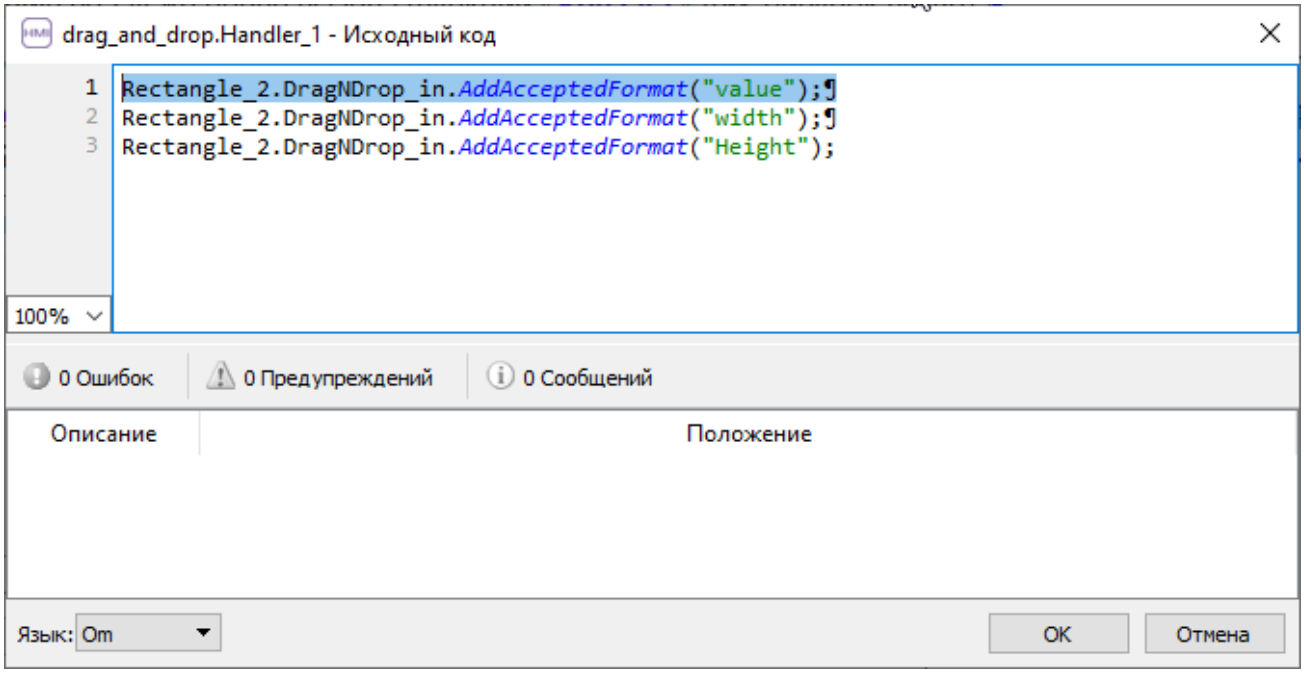
1. Для объекта-отправителя и объекта-получателя добавьте дочерний элемент **Перемещение**.

2. В обработчике события MousePress объекта-отправителя укажите передаваемые данные с помощью функции SetData элемента **Перемещение**. Для каждой отдельной части используйте свою функцию SetData и присваивайте свой тип (идентификатор).

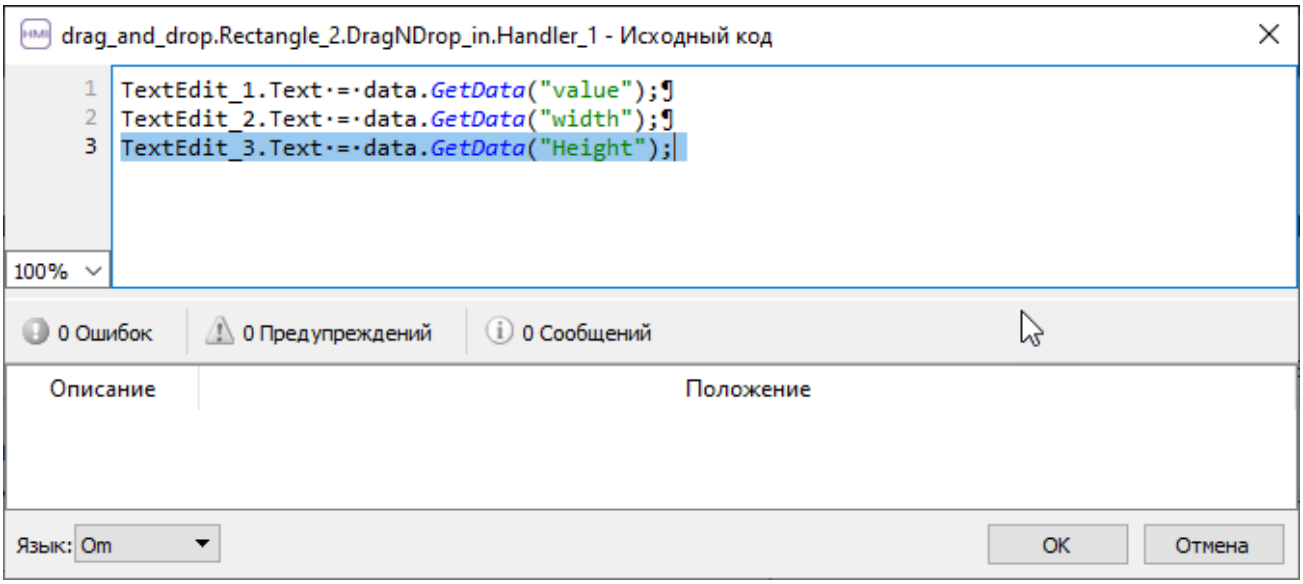


3. Для запуска перемещения данных используйте функцию DoDragDrop элемента **Перемещение**. Вызывайте функцию сразу же после вызова функции SetData (см. рисунок выше).

4. Составьте список разрешённых принимаемых типов для объекта-получателя с помощью функции AddAcceptedFormat элемента **Перемещение**. Необходимо, чтобы список был обработан перед сбросом данных на объект-получатель. К примеру, укажите список принимаемых типов в обработчике события Opened для формы.



5. В обработчике события OnDrop элемента **Перемещение** настройте обработку полученных типов с помощью функции GetData параметра data. Для каждого принимаемого типа используйте свою функцию GetData.

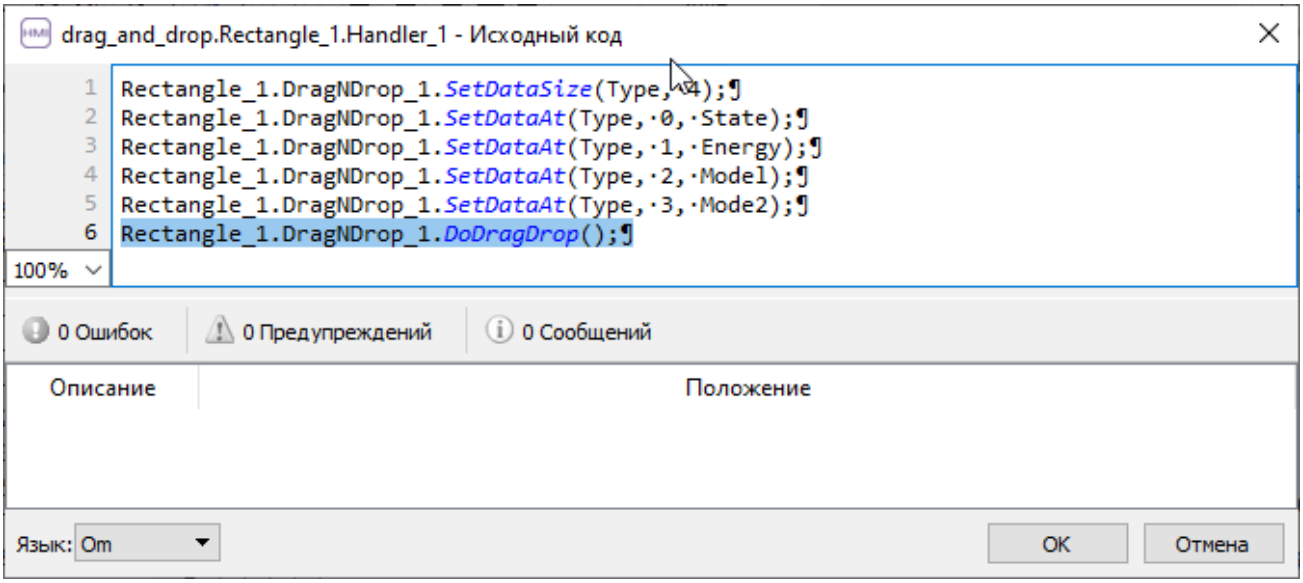


16.3. Передача бинарных данных

Если вы хотите передать блок бинарных данных, передавайте такие данные в виде массива. Имя массива и его размер указывайте с помощью функции SetDataSize. Для указания передаваемых данных используйте функцию SetDataAt элемента **Перемещение** на стороне объекта-отправителя. Для получения размера массива на стороне объекта-получателя используйте функцию GetDataSize параметра data события OnDrop элемента **Перемещение**. Для получения значений из массива используйте функцию GetDataSize параметра data события OnDrop элемента **Перемещение**.

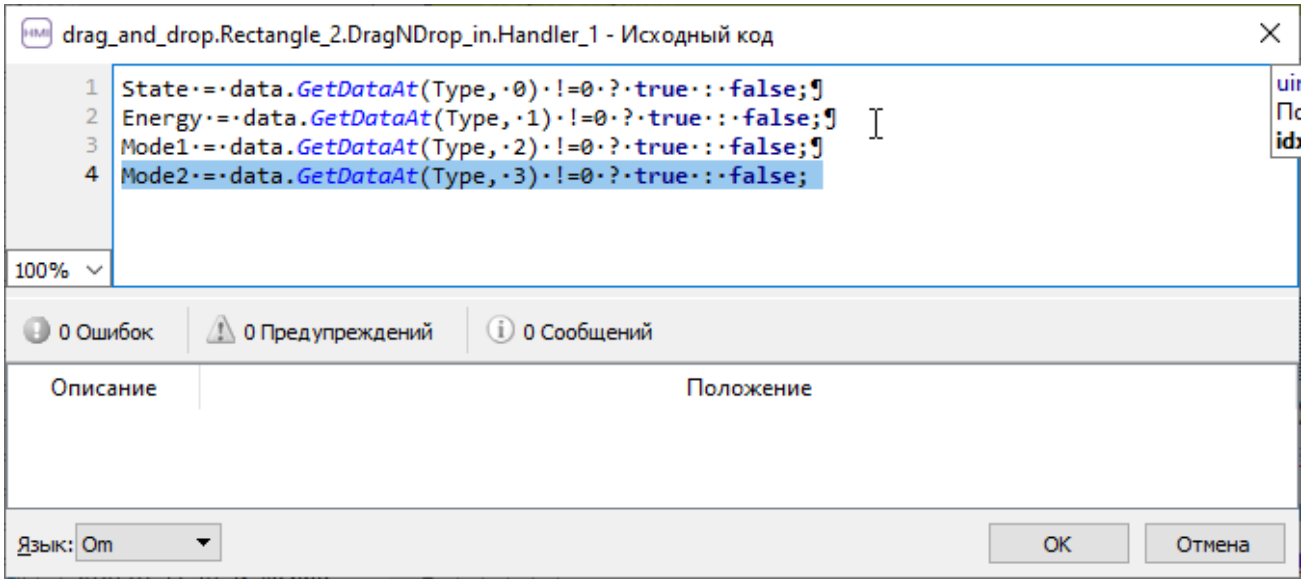
Чтобы настроить передачу данных в виде массива, выполните:

1. Для объекта-отправителя и объекта-получателя добавьте дочерний элемент **Перемещение**.
2. В обработчике события MousePress объекта-отправителя укажите размер и имя массива с помощью функции SetDataSize. Укажите передаваемые данные с помощью функции SetDataAt элемента **Перемещение**. Для каждого элемента массива используйте свою функцию SetDataAt.



3. Для запуска перемещения данных используйте функцию DoDragDrop элемента **Перемещение**. Вызывайте функцию сразу же после вызова функции SetData (см. рисунок выше).

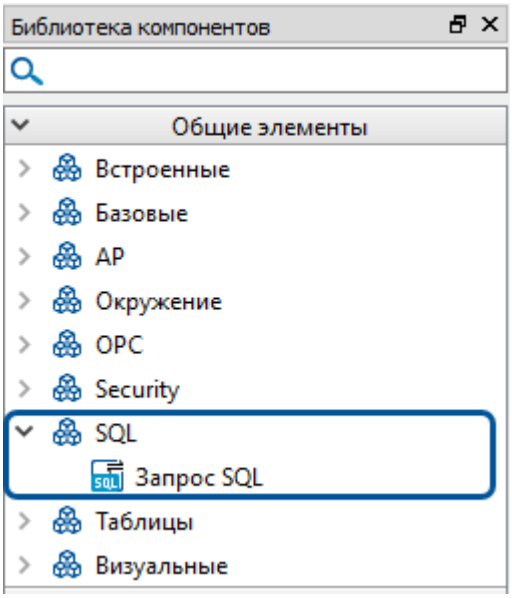
4. На стороне объекта-получателя в обработчике события OnDrop элемента **Перемещение** настройте обработку полученного массива с помощью функции GetDataAt параметра data. Для каждого элемента массива используйте свою функцию GetDataAt.



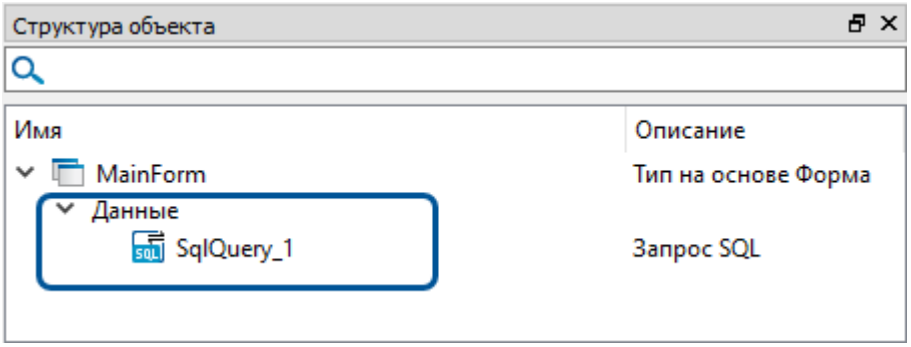
17. Получение данных из БД SQL

Чтобы выполнять SQL-запросы к базам данных для получения, изменения или удаления данных, воспользуйтесь компонентом **Запрос SQL**, который совместим с различными SQL базами данных (PostgreSQL, MySQL, Microsoft SQL Server и другие).

Компонент **Запрос SQL** расположен в разделе **«SQL» библиотеки компонентов**.



Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



Результаты SQL-запросов могут быть представлены в проекте только в виде таблицы, так как таблицы являются основным способом организации и хранения данных в SQL базах данных.

Чтобы отобразить результаты SQL-запросов, подключите компонент **Запрос SQL** к компоненту **Таблица** (стр. 1). Пример настройки указанных компонентов приведен в демонстрационном проекте, описанном ниже.

17.0.1. Как запросить данные из БД PostgreSQL

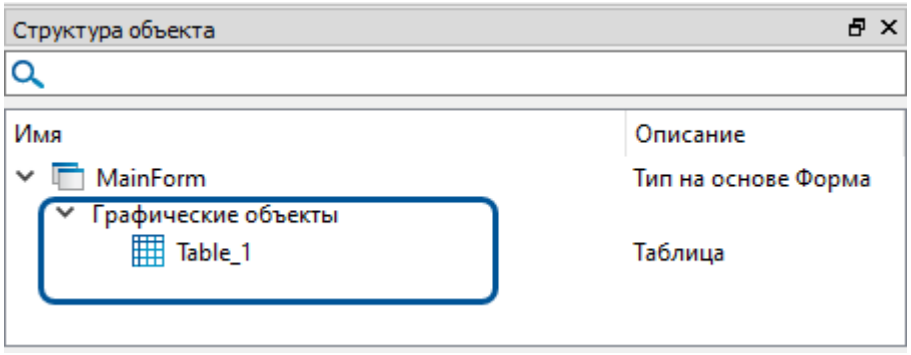
В разделе рассмотрен демонстрационный проект (файлы проекта идут в комплекте с документацией), в котором в качестве примера базы данных используется PostgreSQL, но принципы настройки и использования компонента **Запрос SQL** совпадают для большинства баз данных (PostgreSQL, MySQL, Microsoft SQL Server и другие). Различия могут возникать только в строках подключений к конкретной базе данных и в диалектах текста SQL-запросов.

Проект сделан на языке SePlatform.Om.
В проекте показано как:

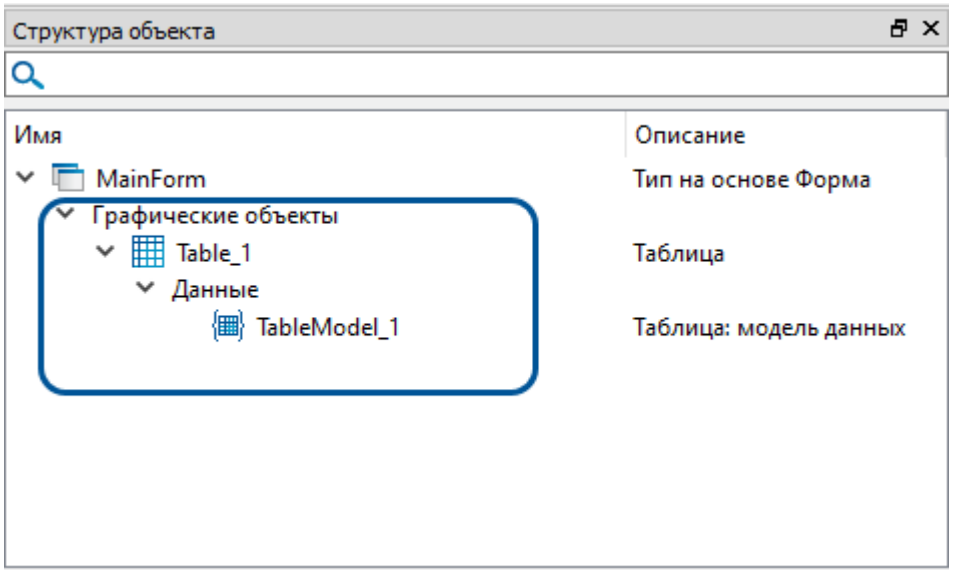
- Подготовить таблицу для вывода результатов SQL-запросов.
- Выводить информацию в текстовое поле о количестве затронутых изменениями строк в базе данных.
- Выполнить SELECT запрос (стр. 198) по нажатию кнопки для получения данных из БД PostgreSQL.
- Разорвать соединение с базой данных после завершения всех операций с ней.

17.0.2. Подготовить таблицу для вывода результатов SQL-запросов

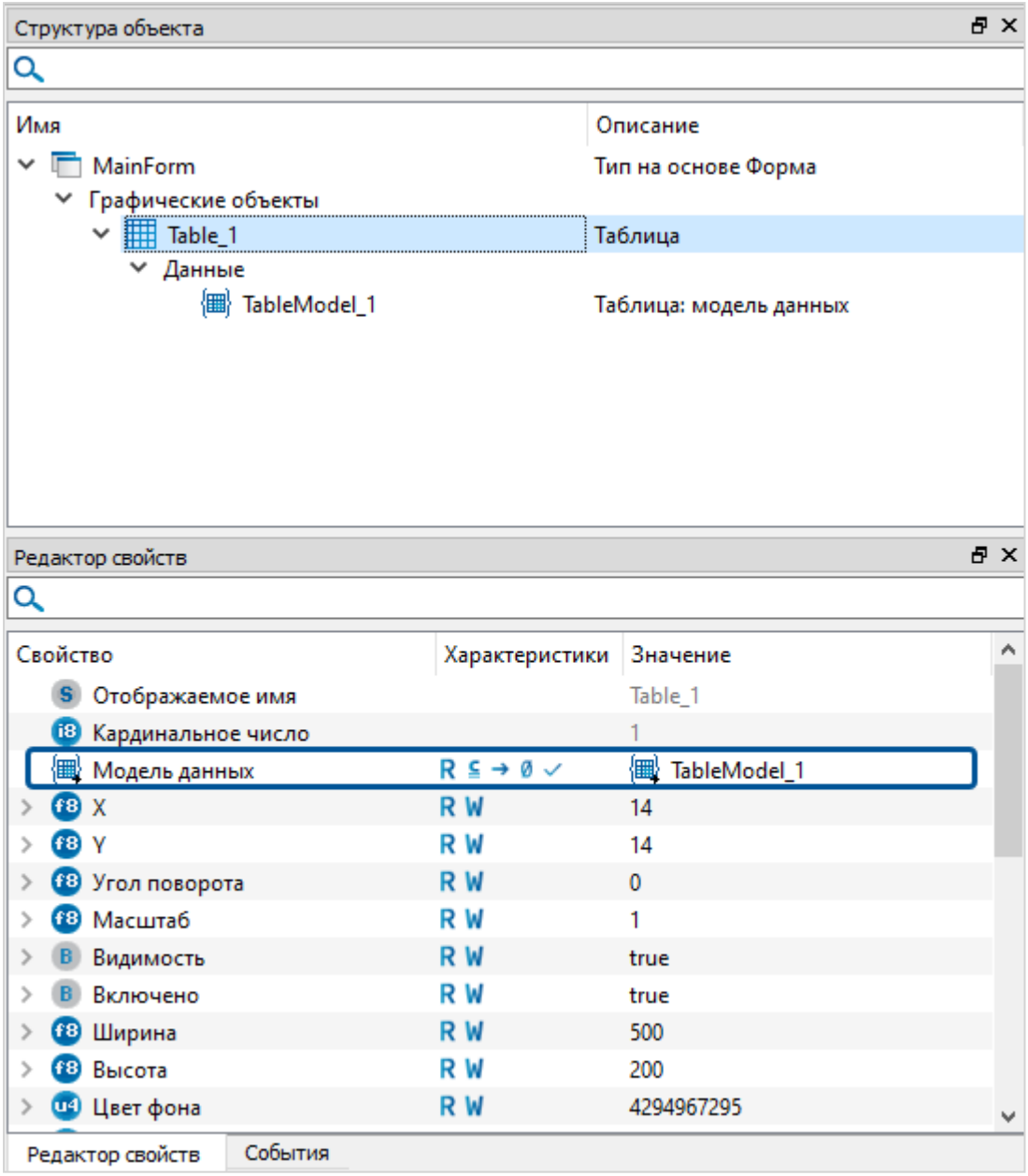
1. Добавьте на экранную форму объект типа Таблица (стр. 108), чтобы иметь возможность помещать данные, полученные из БД, в таблицу.



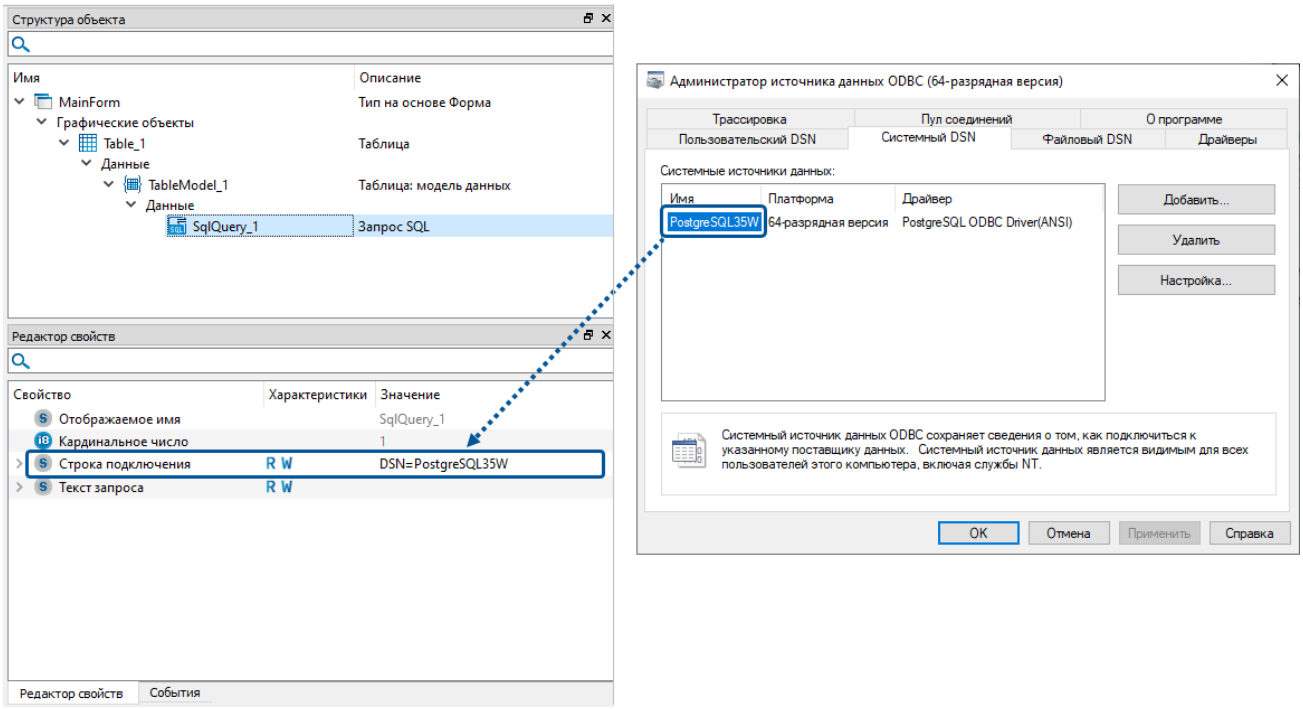
2. Добавьте компонент **Таблица: модель данных** дочерним компоненту **Таблица.**



3. В редакторе свойств у объекта **Таблица** укажите ссылку на добавленный объект **Таблица: модель данных**. Для этого нажмите правой кнопкой мыши по свойству Модель данных → **Сослаться** → **TableModel_1**.



4. Добавьте компонент **Запрос SQL** дочерним компоненту **Таблица: модель данных**. Укажите имя системного источника данных **«DSN»** для вашей базы данных в свойстве Строка подключения. Имя **«DSN»** представляет собой коллекцию параметров, необходимых для подключения к источнику данных ODBC. Системные источники данных ODBC можно посмотреть в **Панель управления** → **Администрирование** → **Администратор источника данных ODBC.**

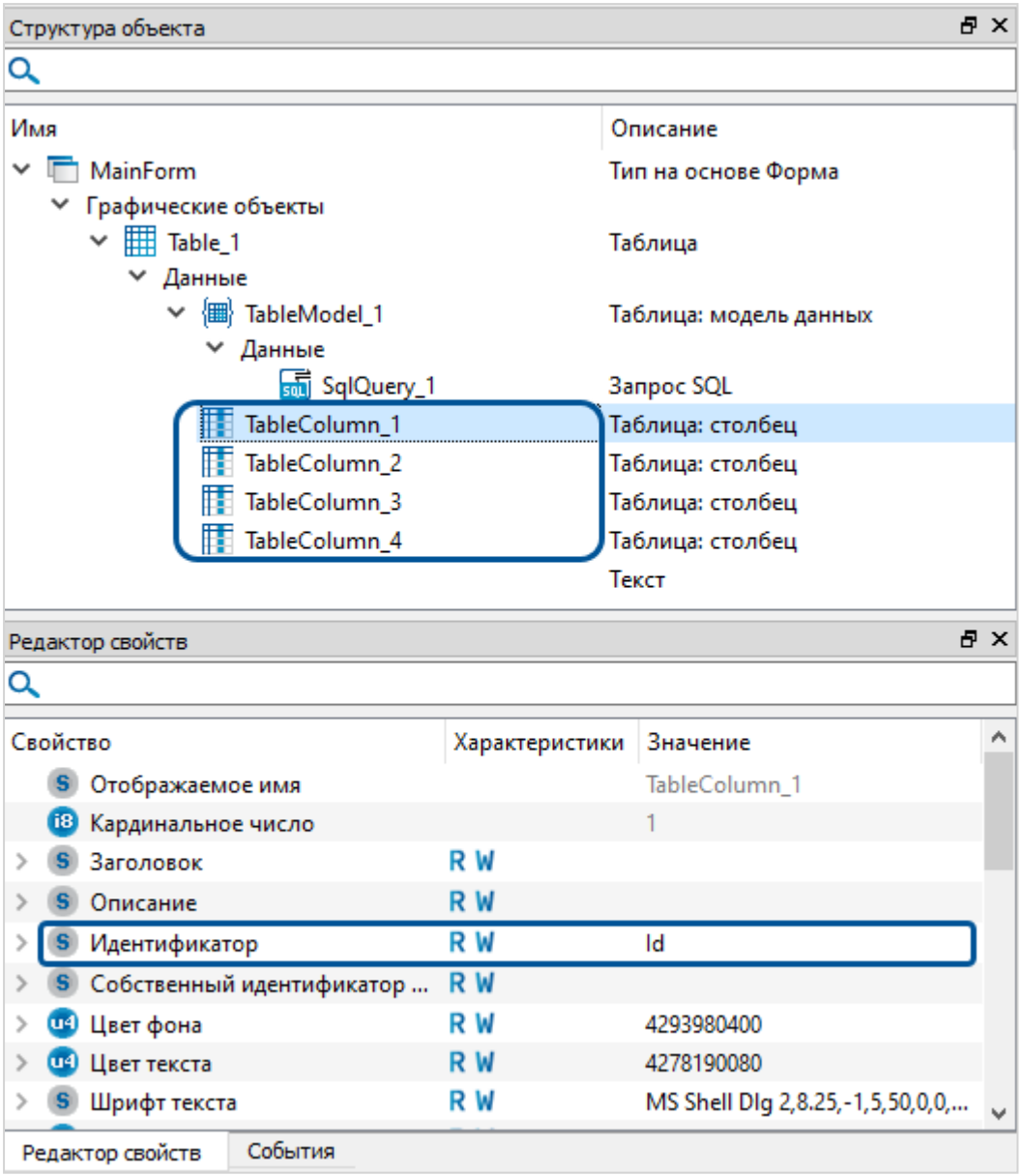


ПРИМЕЧАНИЕ

Для обеспечения соединения с базой данных убедитесь, что для вашей базы данных создан системный источник данных (DSN) в настройках ODBC. В случае отсутствия **«DSN»** для вашей базы данных, вам потребуется создать его.

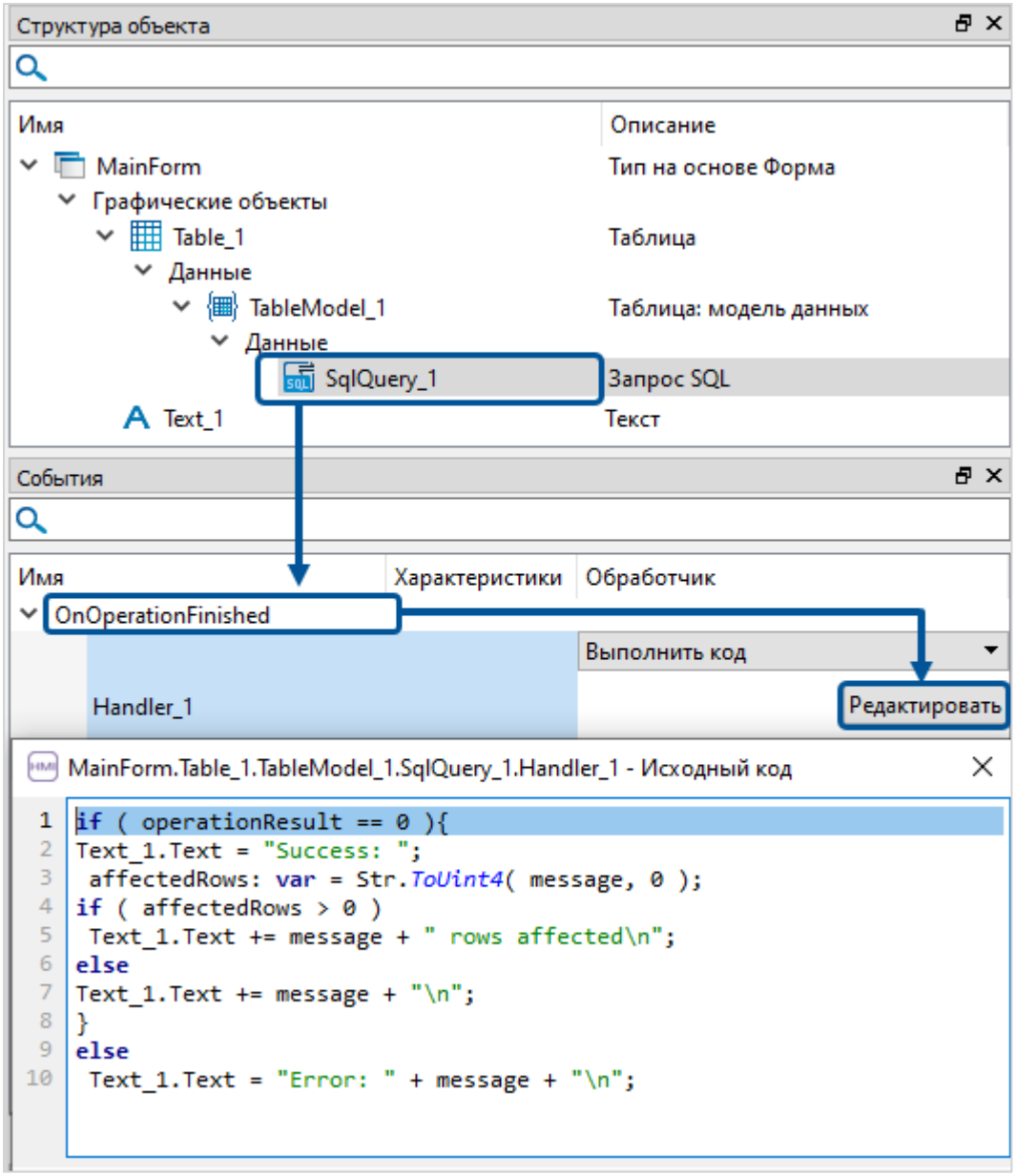
О добавлении базы данных в список системных источников см. в руководстве администратора на модуль **History** (Приложение А).

5. Добавьте компоненту **Таблица** дочерние компоненты **Таблица: столбец**. Вы можете настроить отображение в таблице на форме всех столбцов из базы данных либо только отдельных столбцов. Для каждого объекта **Таблица: столбец** укажите идентификатор столбца в свойстве Идентификатор, который соответствует идентификатору (имени) столбцов в базе данных.



17.0.3. Выводить информацию о количестве затронутых изменениями строк в БД

Чтобы в процессе выполнения SQL запросов к базе данных получать информацию о количестве затронутых изменениями строк, подготовьте текстовое поле и настройте событие OnOperationFinished. В структуре объекта выделите компонент **Запрос SQL**, перейдите во вкладку **События** и нажмите на событие OnOperationFinished правым кликом мыши→ **Добавить обработчик** → **Выполнить код** → **Редактировать**. В открывшемся окне введите следующий код:

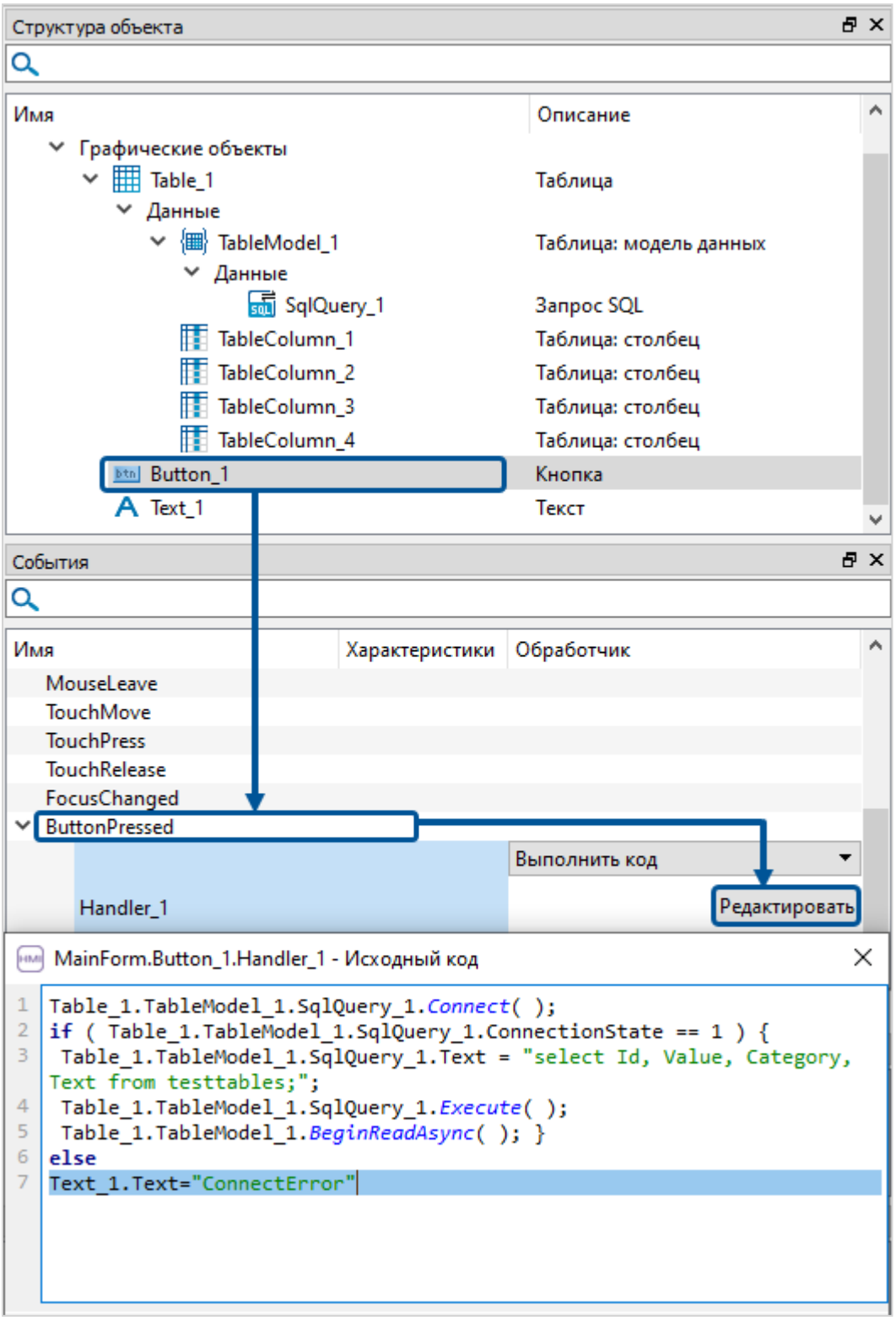


Этот код проверяет результат выполнения операции и выводит информацию о количестве затронутых строк **affectedRows** в базе данных. Если операция выполнена успешно и есть затронутые строки, то выводится сообщение с количеством затронутых строк **«message + rows affected»**. Если операция выполнена успешно, но нет затронутых строк, то выводится только сообщение **«message»**. Если произошла ошибка, то выводится сообщение **«Error: message»**.

```
if ( operationResult == 0 ){
 Text_1.Text = "Success: ";
 affectedRows: var = Str.ToUint4( message, 0 );
if ( affectedRows > 0 )
 Text_1.Text += message + " rows affected\n";
else
 Text_1.Text += message + "\n";
}
else
 Text_1.Text = "Error: " + message + "\n";
```

17.0.4. Выполнить SELECT запрос по нажатию кнопки для получения данных из БД

Добавьте на экранную форму компонент **Кнопка**, при нажатии на которую будет посылаться SQL-запрос к базе данных. Перейдите во вкладку **События**, нажмите на событие ButtonPressed правым кликом мыши→ **Добавить обработчик** → **Выполнить код** → **Редактировать**.



В открывшемся окне установите соединение с базой данных, используя метод Connect(). Задайте текст SQL-запроса с помощью свойства Текст запроса и вызовите метод Execute() для выполнения запроса. Заполните таблицу полученными данными с помощью функции BeginReadAsync():

```
Table_1.TableModel_1.SqlQuery_1.Connect( ); // установка соединения с БД
if ( Table_1.TableModel_1.SqlQuery_1.ConnectionState == 1 ) { // Проверка соединения с базой
данных
 Table_1.TableModel_1.SqlQuery_1.Text = "select Id, Value, Category, Text from testtables;";
// текст SQL-запроса (Выборка значений по столбцам Id, Value, Category, Text из таблицы
testtables)
 Table_1.TableModel_1.SqlQuery_1.Execute( ); // выполнить запрос
 Table_1.TableModel_1.BeginReadAsync( ); } // заполнить таблицу полученными данными из БД
else
 Text_1.Text="ConnectError"
```

ПРИМЕЧАНИЕ

Выполнение SQL-запроса к базе данных можно настроить и в любом другом обработчике события.

17.0.5. Разорвать соединение с базой данных после завершения всех операций с ней

После завершения всех операций с базой данных рекомендуется разрывать соединение с ней. Это действие позволяет эффективно освободить системные ресурсы и избежать потенциальных утечек памяти. Добавьте на экранную форму компонент **Кнопка**, при нажатии на которую будет разрываться соединение с базой данных. В обработчике ButtonPressed пропишите следующий код:

```
Table_1.TableModel_1.SqlQuery_1.Disconnect();
```

Этот код вызовет метод Disconnect() объекта **Запрос SQL**, который разорвет соединение с базой данных.

17.0.6. Примеры выполнения запросов к базе данных

Ниже рассмотрены несколько примеров выполнения базовых SQL-запросов для вставки, изменения, чтения и удаления данных из БД. Запросы к базе данных будут посылаться при нажатии на кнопку (обработчик события ButtonPressed).

ПРИМЕЧАНИЕ

Для выполнения SQL-запросов предварительно добавьте на экранную форму компоненты **Таблица**, **Таблица: модель данных**, **Таблица: столбец**, **Запрос SQL** , **Текст**, **Кнопка** и настройте их, следуя инструкции из предыдущего раздела.

Для избежания возможных проблем, связанных с дублированием идентификаторов столбцов в результирующей выборке SQL-запроса, рекомендуется использовать уникальные идентификаторы столбцов при их выборе. Если в SQL-запросе присутствуют операции объединения таблиц или другие ситуации, когда имена столбцов могут дублироваться, то рекомендуется использовать SQL псевдонимы (временные имена) для столбцов, чтобы обеспечить уникальность идентификаторов.

Пример правильного использования SQL псевдонимов:

Вместо:

```
SELECT * FROM nodes_history h JOIN nodes n ON n.NodeId=h.NodeId WHERE n.TagName
='Test.Parameter' AND (h.time BETWEEN '2023-03-02 07:39:00' AND '2023-03-02 07:39:01')
```

Рекомендуется:

```
SELECT n.nodeid, h.valint, h.quality, h.actualtime FROM nodes_history h JOIN nodes n ON
n.NodeId=h.NodeId WHERE n.TagName ='Test.Parameter' AND (h.time BETWEEN '2023-03-02
07:39:00' AND '2023-03-02 07:39:01')
```

17.0.6.1. Как добавить данные в БД

В этом примере добавляется новая запись в таблицу **«testtables »**для столбцов **«Id »**, **«Value»**, **«Category»**, **«Text»** с указанными значениями.

ПРИМЕР

Добавить в таблицу **«testtables»**новую запись для столбцов **«Id = 1»**, **«Value = 6»**, **«Category = Even»**, **«Text = Six»**

```
Table_1.TableModel_1.SqlQuery_1.Connect();// Установка соединения с базой данных
if ( Table_1.TableModel_1.SqlQuery_1.ConnectionState == 1 ) { // Проверка соединения
с базой данных
 Table_1.TableModel_1.SqlQuery_1.Text = "INSERT INTO testtables (Id, Value, Category,
Text) VALUES (1, 6, 'Even', 'Six')"; // Установка текста SQL-запроса
 Table_1.TableModel_1.SqlQuery_1.Execute( ); } // Выполнение запроса INSERT
else
 Text_1.Text="ConnectError"
```

17.0.6.2. Как обновить данные в БД

В этом примере обновляется значение столбца **«Value»** в строке, где значение столбца **«Id»** равно **«2»**.

ПРИМЕР

Обновить в таблице **«testtables»**значение столбца **«Value»** на **«343»**, где **«Id = 2»**.

```
Table_1.TableModel_1.SqlQuery_1.Connect();// Установка соединения с базой данных
if ( Table_1.TableModel_1.SqlQuery_1.ConnectionState == 1 ) { // Проверка соединения
с базой данных
 Table_1.TableModel_1.SqlQuery_1.Text = "UPDATE testtables SET Value = 343 WHERE Id =
2"; // Установка текста SQL-запроса
 Table_1.TableModel_1.SqlQuery_1.Execute( ); } // Выполнение запроса UPDATE
else
 Text_1.Text="ConnectError"
```

17.0.6.3. Как получить данные из БД

В этом примере получены все данные из таблицы **«testtables»**.

ПРИМЕР

Получить значения всех столбцов таблицы**«testtables»**.

```
Table_1.TableModel_1.SqlQuery_1.Connect();// Установка соединения с базой данных
if ( Table_1.TableModel_1.SqlQuery_1.ConnectionState == 1 ) { // Проверка соединения
с базой данных
 Table_1.TableModel_1.SqlQuery_1.Text = "SELECT * from testtables"; // Установка
текста SQL-запроса
 Table_1.TableModel_1.SqlQuery_1.Execute( ); // Выполнение запроса SELECT
 Table_1.TableModel_1.BeginReadAsync( ); } // Полученными данными заполняем таблицу
else
 Text_1.Text="ConnectError"
```

17.0.6.4. Как удалить данные из БД

В этом примере удаляются все строки из таблицы **«testtables»**, где значение столбца **«Category»** равно **«Even»**.

ПРИМЕР

Удалить все строки из таблицы **«testtables»**, где значение столбца **«Category»** равно **«Even»**.

```
Table_1.TableModel_1.SqlQuery_1.Connect();// Установка соединения с базой данных
if ( Table_1.TableModel_1.SqlQuery_1.ConnectionState == 1 ) { // Проверка соединения
с базой данных
 Table_1.TableModel_1.SqlQuery_1.Text = "DELETE FROM testtables WHERE Category =
'Even'"; // Установка текста SQL-запроса
 Table_1.TableModel_1.SqlQuery_1.Execute( ); } // Выполнение запроса DELETE
else
 Text_1.Text="ConnectError"
```

Это всего лишь несколько примеров сценариев работы с компонентом **Запрос SQL** в SePlatform.HMI. Вы можете создавать более сложные запросы, использовать условия, соединять таблицы и многое другое в зависимости от ваших потребностей.

17.0.7. Настройка переменных окружения для взаимодействия с SQL базой данных на Linux

При взаимодействии с SQL базой данных через SePlatform.HMI на операционных системах семейства Linux необходимо предварительно настроить переменные окружения **«ODBCSYSINI»** и **«LD_LIBRARY_PATH»**. Эти переменные окружения необходимы для того, чтобы система знала, где искать конфигурационные файлы для корректной работы с SQL базой данных через ODBC (интерфейс между базой данных и приложением, взаимодействующим с ней).

Самый распространенный способ настройки переменных окружения на Linux- это добавить переменные в системный файл `.bashrc`.
Для добавления переменных окружения в файл `.bashrc` откройте терминал и выполните следующие команды:

```
# Эта команда откроет файл .bashrc для редактирования.
nano ~/.bashrc
# Эта строка устанавливает переменную ODBCSYSINI в значение "/etc".
export ODBCSYSINI="/etc"
# Эта строка устанавливает переменную LD_LIBRARY_PATH в значение "/usr/lib/x86_64-linux-
gnu/odbc".
export LD_LIBRARY_PATH="/usr/lib/x86_64-linux-gnu/odbc"
```

Сохраните файл, нажав Ctrl + X, затем Y, затем Enter.

Обновите текущий сеанс оболочки, чтобы изменения вступили в силу:

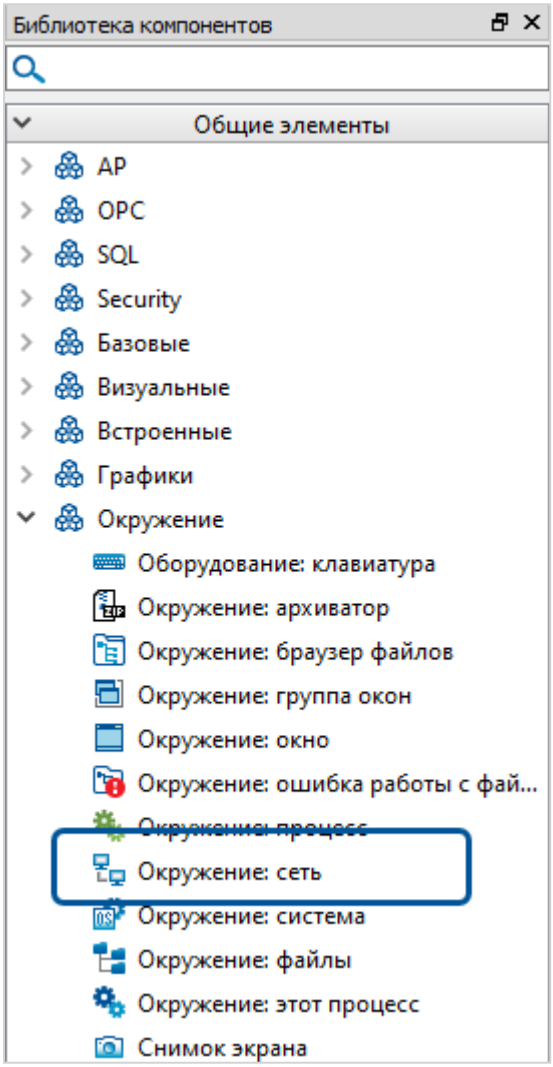
```
source ~/.bashrc
```

18. Работа с файлами и оборудованием

Чтобы взаимодействовать с файловым окружением и оборудованием, воспользуйтесь набором компонентов SePlatform.HMI для взаимодействия с сетью, файловой системой и оборудованием компьютера.

18.1. Сетевое окружение

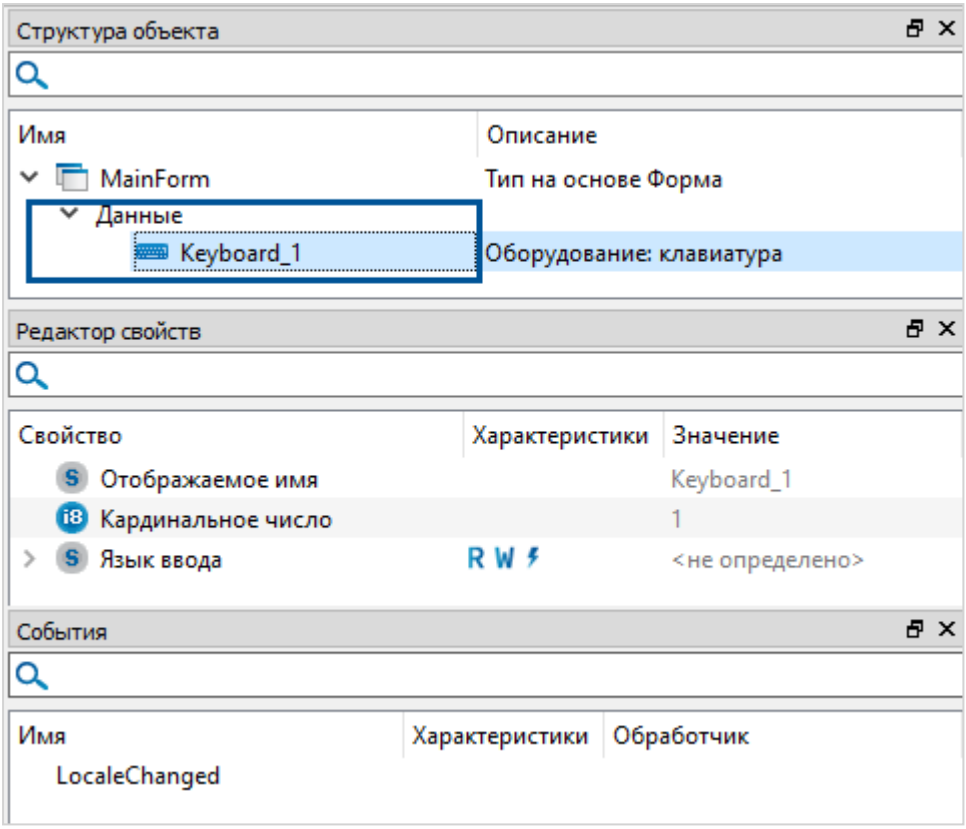
Чтобы взаимодействовать с сетевым окружением, добавьте на экранную форму компонент **Окружение: сеть**.



Компонент позволяет получить имя сетевого компьютера через свойство ComputerName.

18.2. Клавиатура

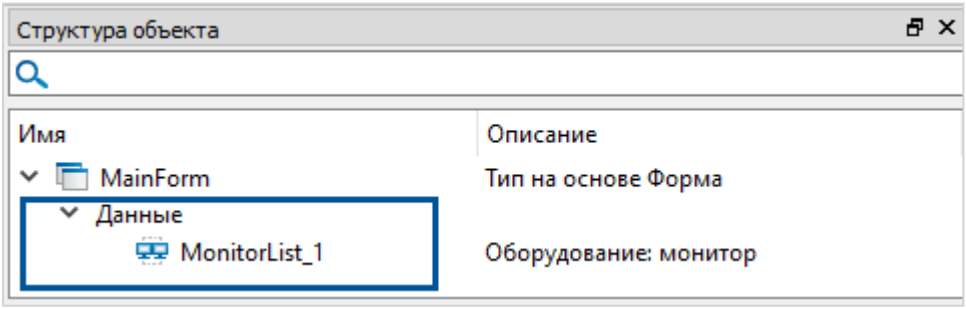
Чтобы взаимодействовать с клавиатурой, добавьте на экранную форму компонент **Оборудование: клавиатура**. Компонент невизуальный и виден только в области **Структура объекта**.



Компонент позволяет получить информацию о текущей раскладке клавиатуры (свойство Locate), состояниях клавиш и обрабатывать события смены раскладки (LocaleChanged).

18.3. Монитор

Чтобы взаимодействовать с монитором (или группой мониторов многомониторной системы отображения), добавьте на экранную форму компонент **Оборудование: монитор**. Компонент невизуальный и виден только в области **Структура объекта**.



Компонент позволяет получить доступ к функциям взаимодействия с мониторами.

18.4. Запуск внешних приложений

Для запуска внешних приложений из проекта SePlatform.HMI используются следующие компоненты:

- **Окружение: процесс** позволяет запускать внешние исполняемые файлы с определенными аргументами командной строки.
- **Окружение: этот процесс** предоставляет информацию о текущем процессе SePlatform.HMI, включая идентификатор процесса, путь к исполняемому файлу и рабочий каталог процесса.

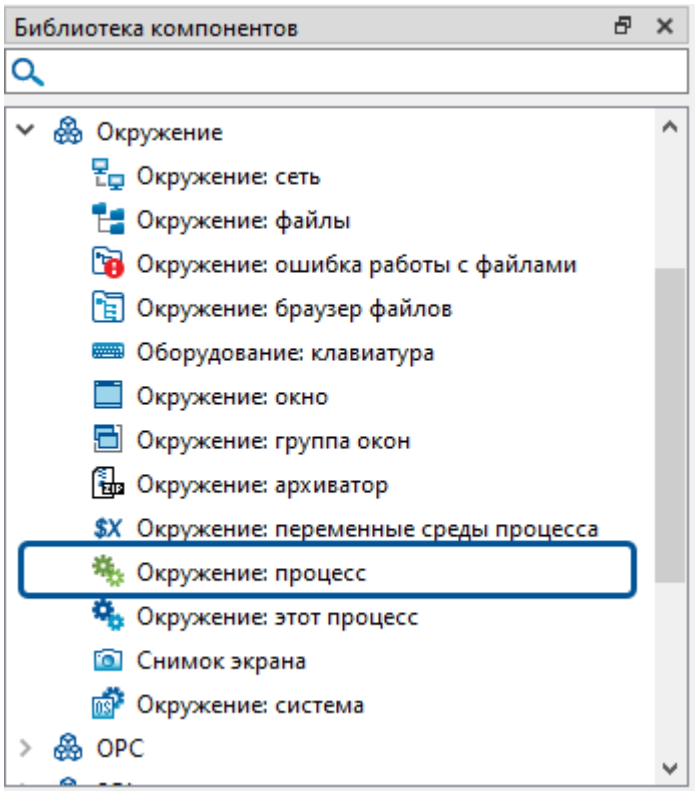
- **Окружение: переменные среды процесса** позволяет настраивать переменные окружения для запускаемых процессов.

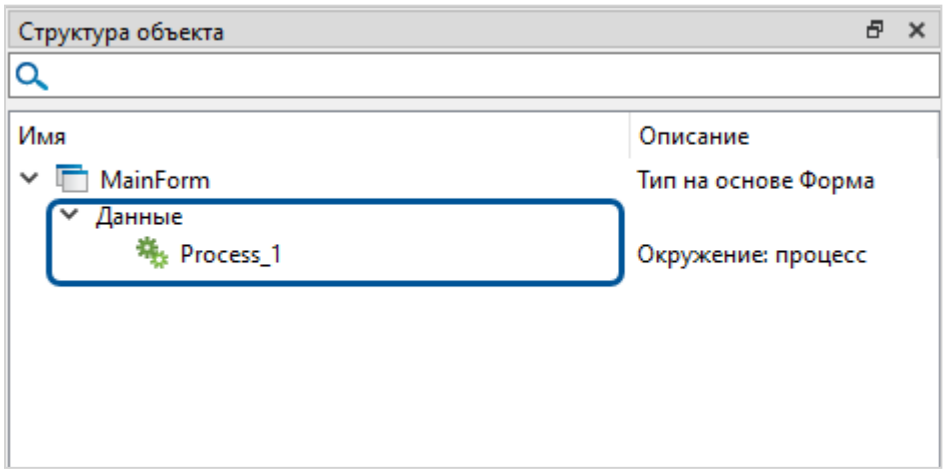
Для получения подробной информации о настройке данных компонентов, а также примерах их использования, обратитесь к демонстрационному проекту.

18.4.1. Окружение: процесс

Чтобы запускать внешние исполняемые файлы (приложения), добавьте на экранную форму компонент **Окружение: процесс**. Компонент предоставляет возможность запускать различные приложения прямо из вашего проекта SePlatform.HMI. Вы можете указать, какое приложение запустить и передать ему нужные параметры запуска из кода обработчика.

Компонент **Окружение: процесс** расположен в юните **«Окружение»** библиотеки компонентов. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



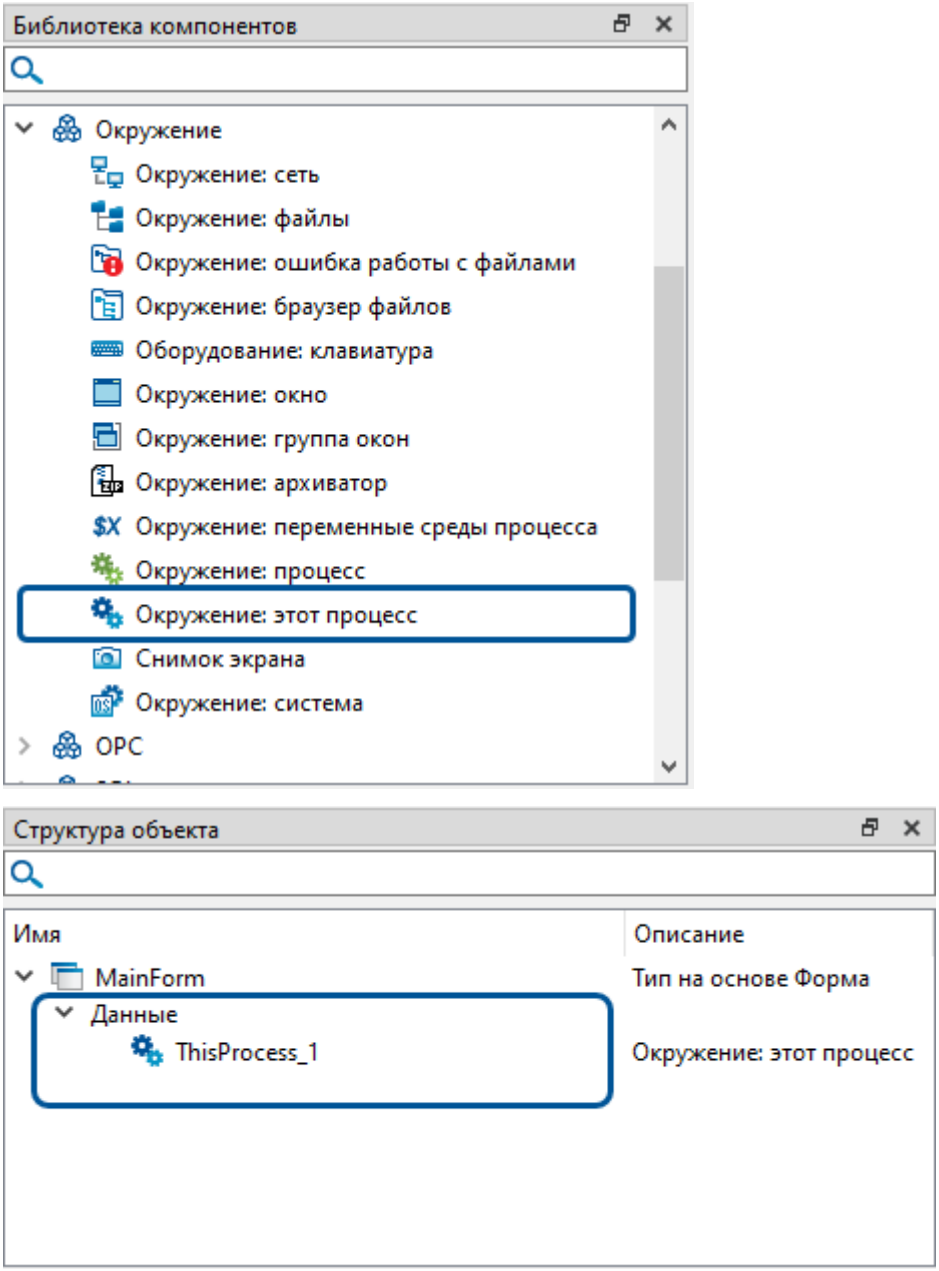


Свойства, методы и события компонента **Окружение: процесс** рассмотрены в справочном руководстве.

18.4.2. Окружение: этот процесс

Чтобы получать информацию о текущем процессе SePlatform.HMI , добавьте на экранную форму компонент **Окружение: этот процесс**.С помощью этого компонента вы можете получить идентификатор процесса SePlatform.HMI, путь к его исполняемому файлу и рабочему каталогу процесса.

Компонент **Окружение: этот процесс** расположен в юните **«Окружение»** библиотеки компонентов. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.

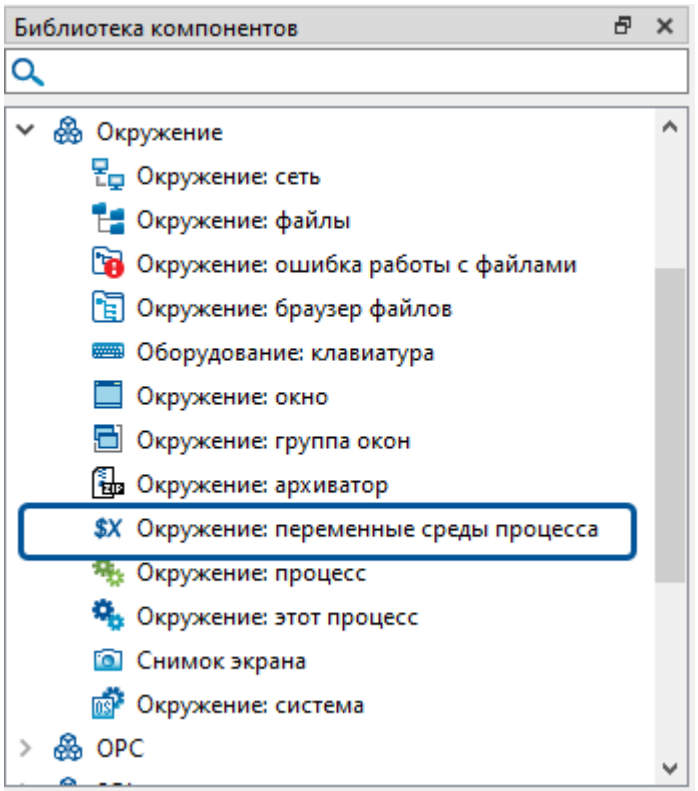


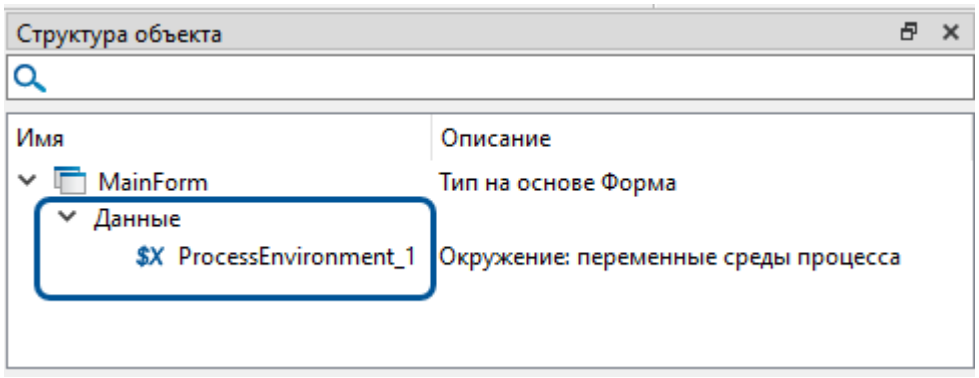
Свойства, методы и события компонента **Окружение:этот процесс** рассмотрены в справочном руководстве.

18.4.3. Окружение: переменные среды процесса

Переменные среды - это специальные значения, которые хранятся в операционной системе и используются для передачи информации между различными процессами и программами. Переменные содержат данные, такие как пути к исполняемым файлам, параметры конфигурации, настройки системы и многое другое.

Для просмотра системных переменных в операционной системе Windows, перейдите в: **Панель управления** -> **Система и безопасность** -> **Система** -> **Дополнительные параметры системы** -> **Вкладка "Дополнительно"** -> **Кнопка "Переменные среды"**.
Компонент **Окружение: переменные среды процесса** предоставляет вам возможность настраивать и управлять переменными среды для запускаемых процессов в вашем проекте. Это позволяет вам задавать свои собственные значения переменных или получать информацию о системных переменных среды, которые будут использоваться при запуске процессов с помощью компонента **Окружение: процесс**.
Компонент **Окружение: переменные среды процесса** расположен в юните **«Окружение»** библиотеки компонентов. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.





Свойства и методы компонента **Окружение: переменные среды процесса** рассмотрены в справочном руководстве.

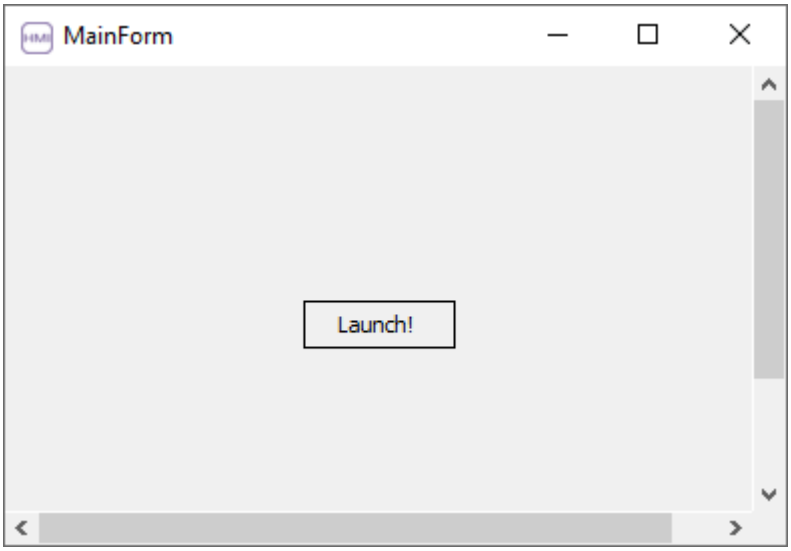
18.4.4. Как настроить запуск внешнего приложения (Демо-проект)

Далее кратко описан небольшой пример (файлы проекта идут в комплекте с документацией) использования компонента **Окружение: процесс** в проекте SePlatform.HMI с использованием языка SePlatform.Om.

В примере показано, как:

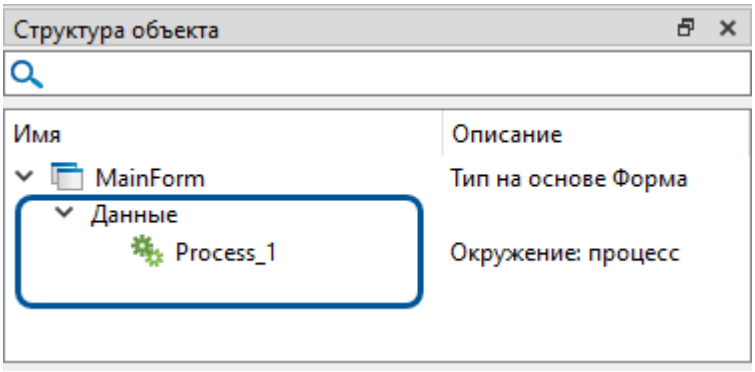
- Запустить скрипт Python из проекта SePlatform.HMI с определенными параметрами (Путь к исполняемому файлу и Аргументы запуска).
- Указать конкретную версию Python для запуска скрипта, используя системную переменную PATH.

В результате будет подготовлена форма, позволяющая по нажатию кнопки запускать скрипт Python из проекта SePlatform.HMI с нужной версией Python.



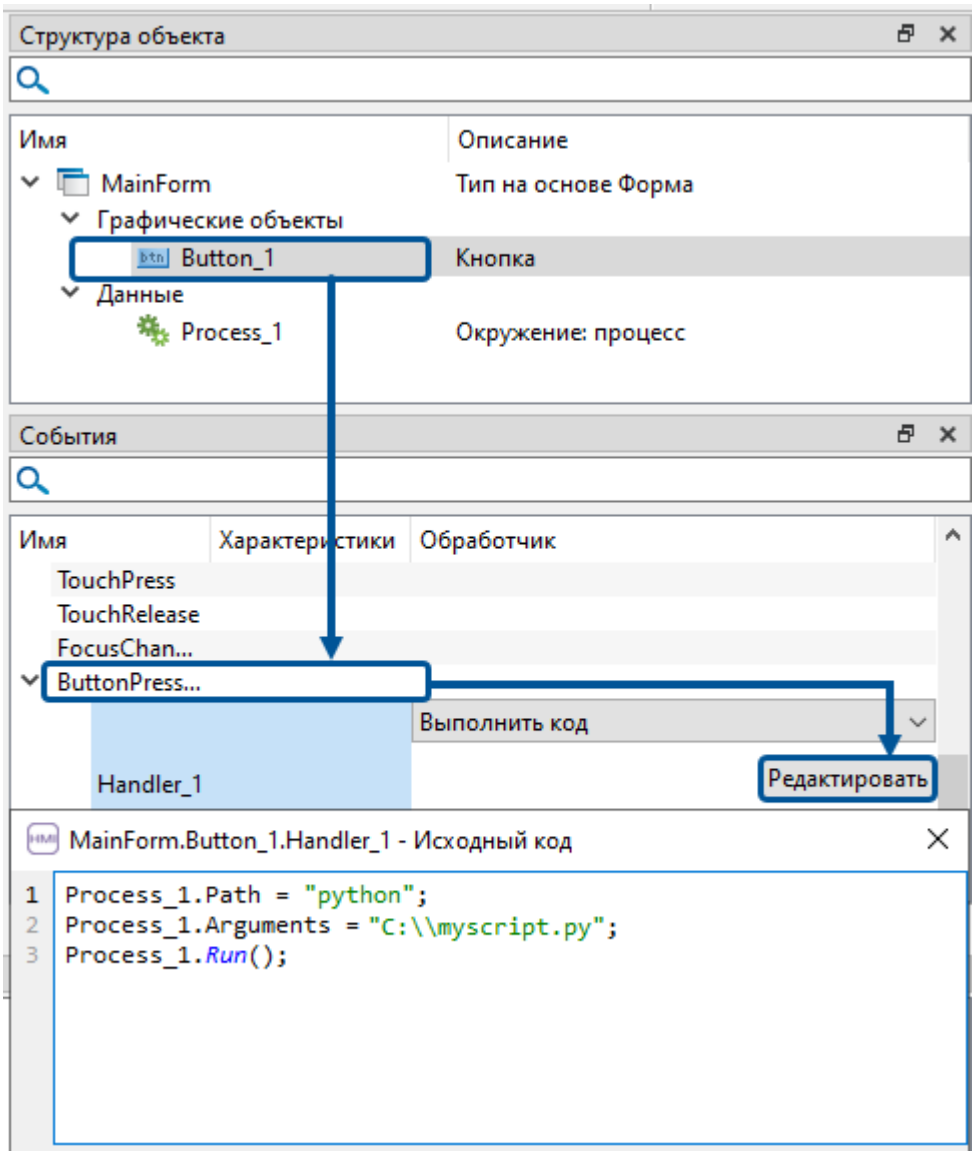
18.4.4.1. Запуск внешнего приложения из проекта SePlatform.HMI

1. Переместите файл скрипта `myscript.py` в корневой каталог диска C.
2. Чтобы запускать скрипт Python из проекта SePlatform.HMI, добавьте на форму компонент **Окружение: процесс**.



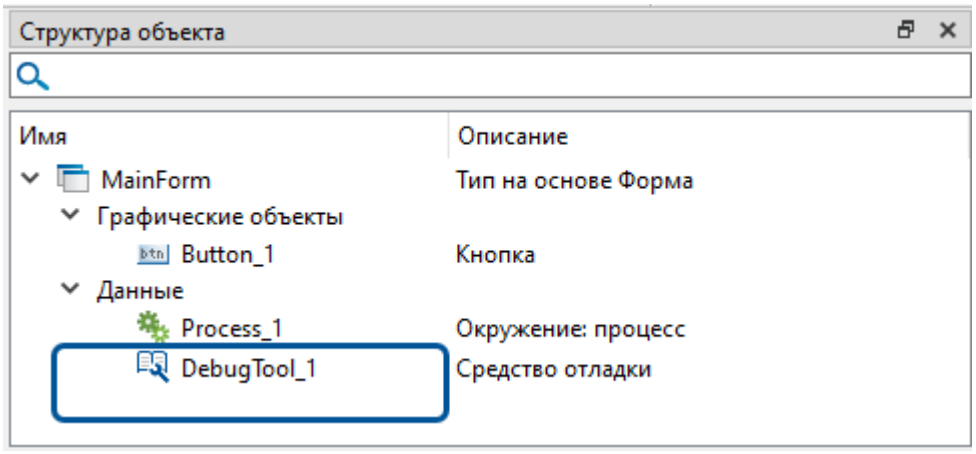
3. Добавьте на форму компонент **Кнопка**, по нажатию которой будет происходить запуск скрипта Python с заданными параметрами. Для задания пути к исполняемому файлу будет использоваться свойство **Путь к исполняемому файлу**, а для задания аргументов запуска скрипта Python будет использоваться свойство **Аргументы запуска**.

В обработчике **ButtonPressed** пропишите следующий код:




```
Process_1.Path = "python"; // Указываем путь к исполняемому файлу
Process_1.Arguments = "C:\\myscript.py"; // Указываем параметры запуска
Process_1.Run(); // Запускаем процесс
```

4. Для того чтобы наблюдать вывод строк, генерируемых запущенным Python-скриптом, добавьте на форму компонент **Средство отладки**.



5. Чтобы строки вывода автоматически записывались в журнал исполнения, в обработчике события OnStdout компонента **Окружение: процесс** пропишите следующий код:

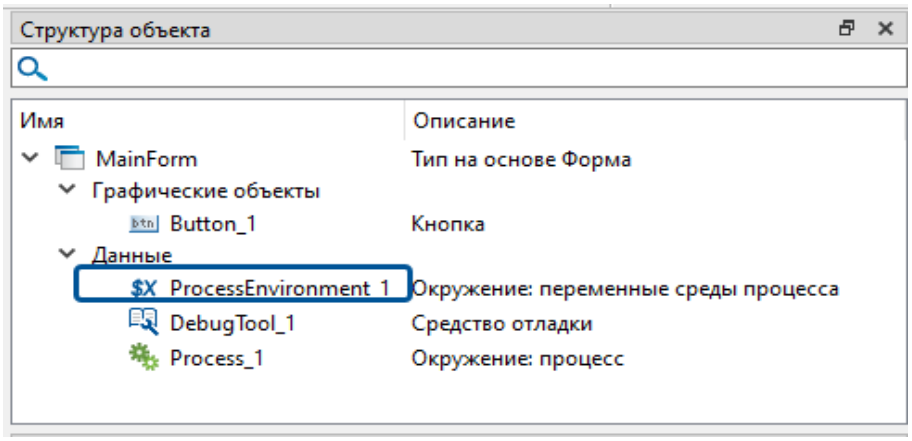
```
DebugTool_1.Log(value);
```

18.4.4.2. Настройка пользовательских переменных среды

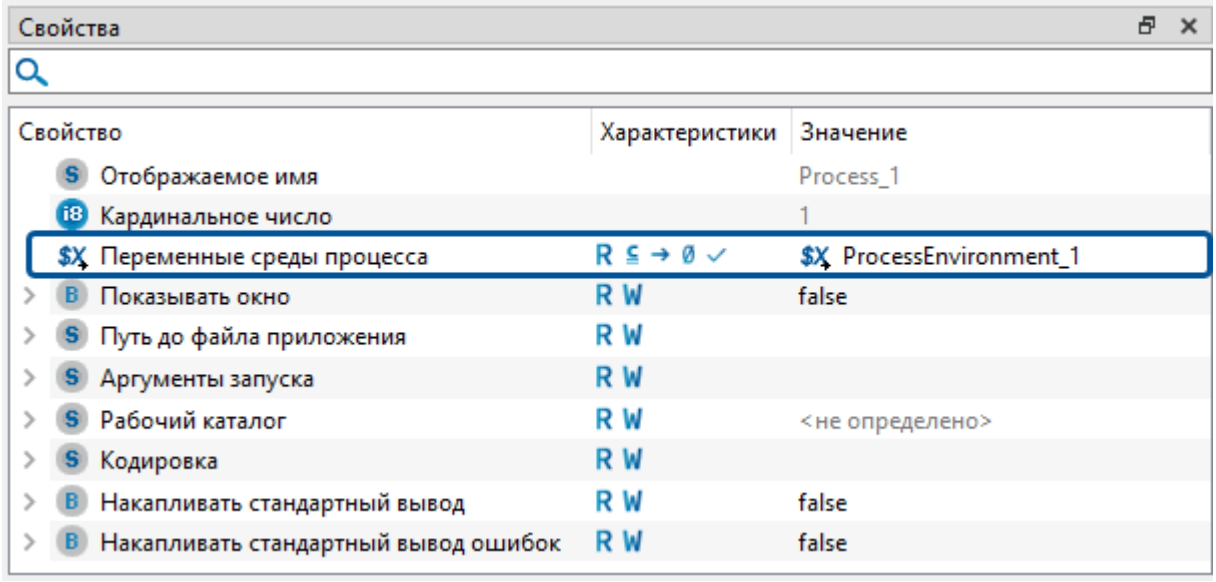
В следующем примере показано, как через переменную среды PATH указать конкретную версию Python при запуске скрипта. PATH - это системная переменная среды, которая содержит список путей к папкам, где система ищет исполняемые файлы при их вызове без указания полного пути. Настройка переменной PATH позволяет нам указать системе, где найти исполняемый файл Python, обеспечивая доступ к определенной версии Python.

Для настройки переменной воспользуемся компонентом **Окружение: переменные среды процесса** и его функцией Prepend().

1. Добавьте на форму компонент **Окружение: переменные среды процесса**.



2. У объекта **Окружение: процесс** для свойства Переменные среды процесса укажите в значении ссылку на добавленный объект типа **Окружение: переменные среды процесса**. Для это нажмите правой кнопкой мыши по свойству Переменные среды процесса → **Сослаться**→ **ProcessEnvironment_1**.



3. Для настройки переменных среды процесса и последующего запуска скрипта с нужной версией Python, в обработчике ButtonPressed для компонента **Кнопка** пропишите следующий код:

```
ProcessEnvironment_1.Prepend("Path" , "C:\\Python39"); // Настраиваем переменную PATH
для запускаемого скрипта

Process_1.Path = "python"; // Указываем путь к исполняемому файлу
Process_1.Arguments = "C:\\myscript.py"; // Указываем параметры запуска
Process_1.Run(); // Запускаем процесс
```

Этот код добавляет путь к установленной версии Python в переменной PATH перед запуском скрипта. Таким образом, система будет использовать указанную версию Python для выполнения скрипта.

18.5. Просмотр файловой системы

Для просмотра файловой системы вам понадобятся следующие компоненты:

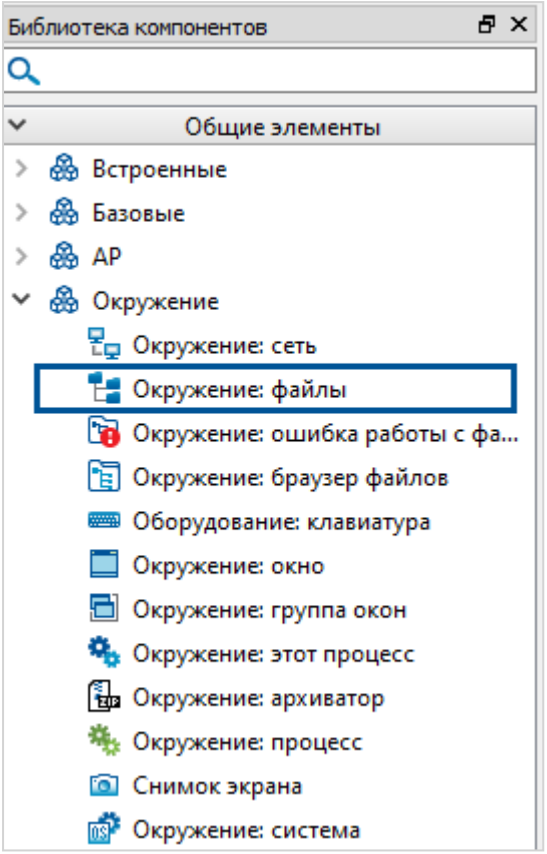
- Компонент **Окружение: файлы**, который используется для работы с каталогами файловой системы, чтения и записи содержимого файлов.
- Компонент **Окружение: ошибка работы с файлами**, который позволяет отслеживать возникающие ошибки при выполнении операций с файловой системой.
- Компонент **Окружение: браузер файлов**, который позволяет просматривать содержимое файловой системы.

Для получения подробной информации о настройке данных компонентов, а также примерах их использования, обратитесь к демонстрационному проекту (стр. 214).

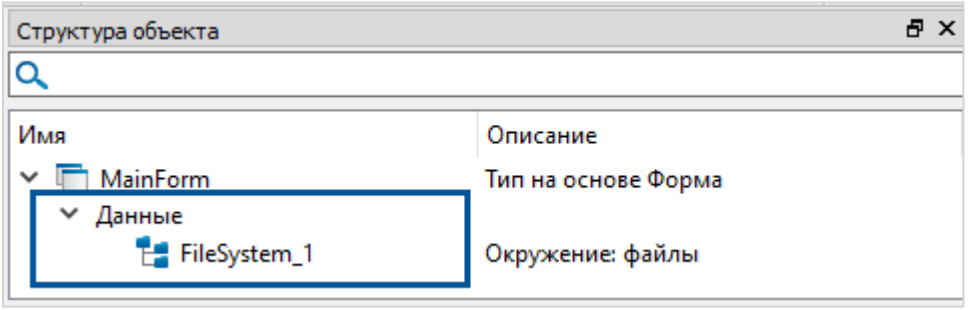
18.5.1. Файлы

Чтобы работать с каталогами файловой системы, читать и записывать содержимое файлов, добавьте на экранную форму компонент **Окружение: файлы**.

Компонент **Окружение: файлы** расположен в юните **«Окружение» библиотеки компонентов**.



Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.

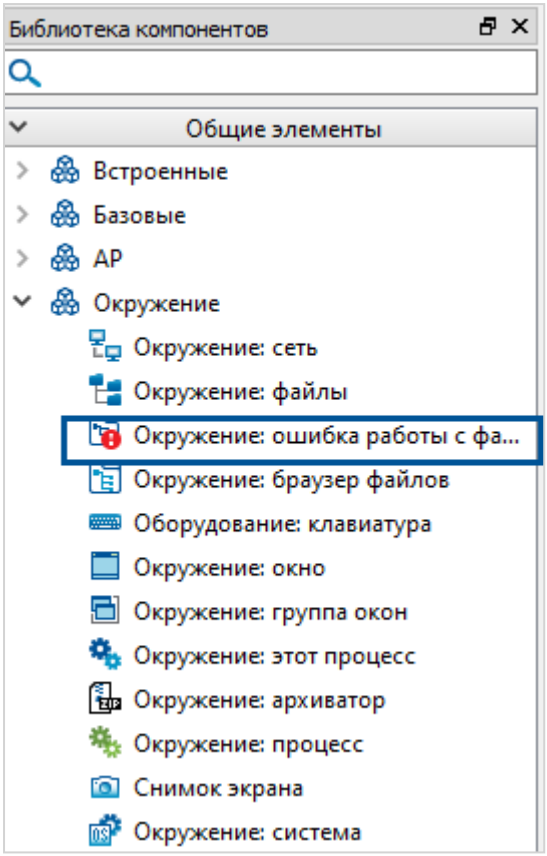


С помощью данного компонента вы можете проверять существование файлов и папок, управлять путями и корневым путем, создавать новые папки, а также читать и записывать содержимое текстовых файлов.

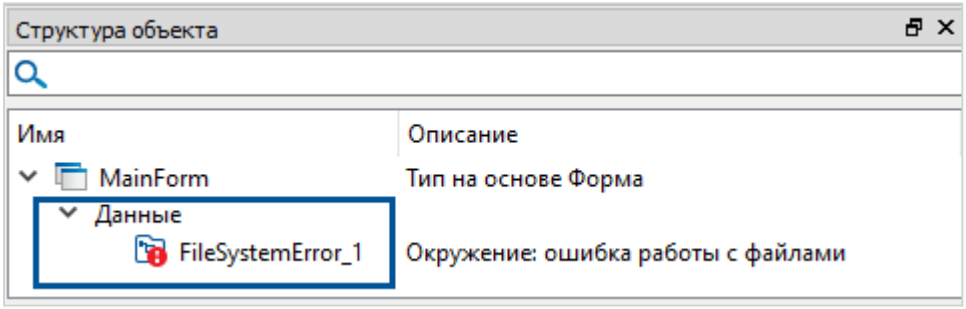
18.5.2. Ошибка работы с файлами

Чтобы отслеживать ошибки, возникающие при работе с файловой системой, добавьте на экранную форму компонент **Окружение: ошибка работы с файлами**.

Компонент **Окружение: ошибка работы с файлами** расположен в юните **«Окружение» библиотеки компонентов**.



Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.

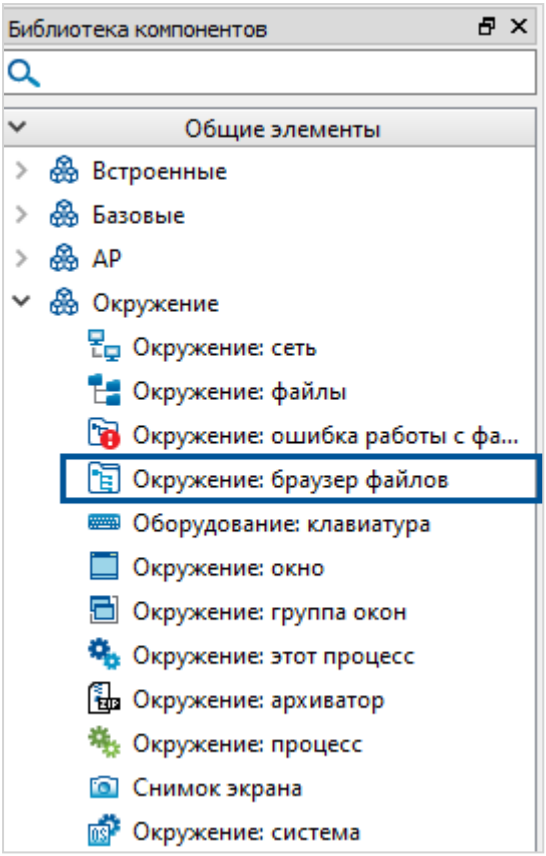


С помощью этого компонента вы сможете получить подробные сведения о возникших ошибках.

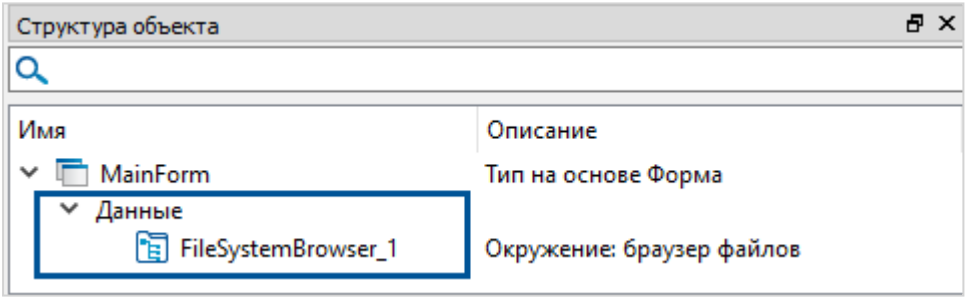
18.5.3. Браузер файлов

Чтобы просматривать содержимое файловой системы, добавьте на экранную форму компонент **Окружение: браузер файлов**.

Компонент **Окружение: браузер файлов**расположен в юните **«Окружение» библиотеки компонентов**.



Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



С помощью этого компонента вы сможете настраивать отображение файлов и папок.
Для просмотра содержимого файловой системы подключите **Окружение: браузер файлов** к одному из компонентов: **Дерево** или **Таблица**.

18.5.3.1. Какие данные возвращает компонент

Компонент **Окружение: браузер файлов** возвращает данные в табличном формате. Используя указанные ниже идентификаторы, вы можете обращаться к этим данным в проекте. Например, вывести их значения в виде древовидной структуры на форме, используя компоненты **Источник данных дерева**, **Дерево** и **Колонка дерева** из юнита **«Визуальные»**.

Идентификатор колонки	Тип колонки	Тип возвращаемого значения	Описание
name	1	string	Имя элемента.
id	2	string	Идентификатор элемента.
parent_id	3	string	Идентификатор родителя.

type	5	uint1	Тип элемента файловой системы › **«0»** - директория; › **«1»** - файл; › **«2»** - ссылка.
ext	5	string	Расширение файла.
size	5	uint8	Размер файла.
created	5	timestamp	Время создания.
modified	5	timestamp	Время изменения.
expandable	6	bool	Возможность раскрыть элемент и просмотреть его содержимое. › **«true»**- отображение содержимого элемента доступно; › **«false»** - отображение содержимого элемента недоступно.

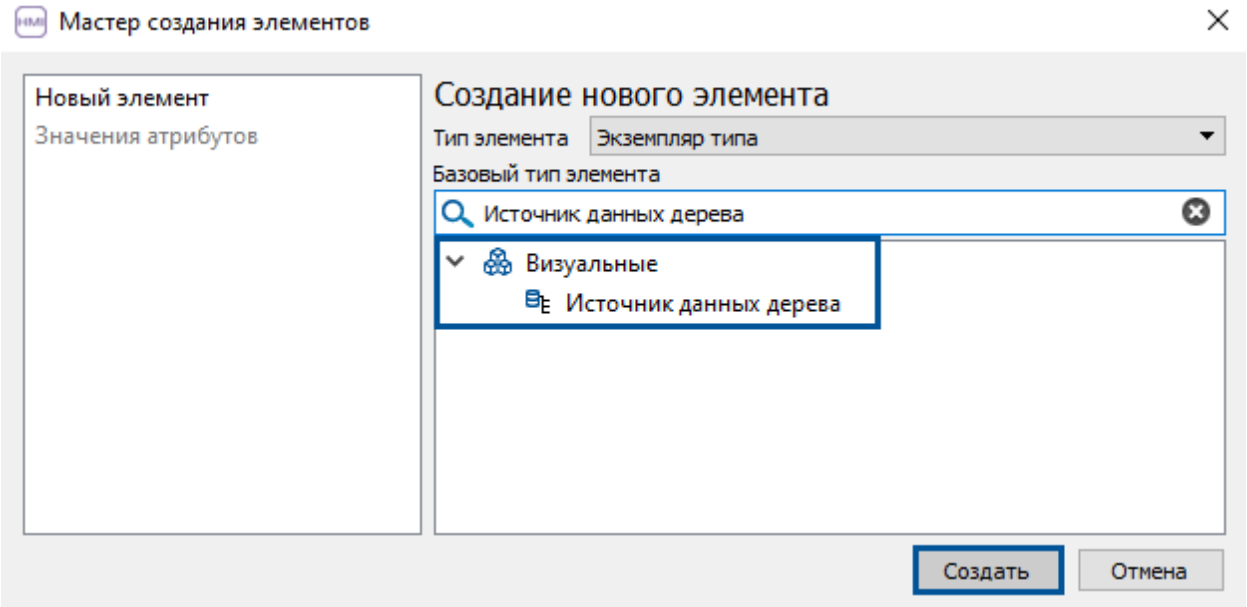
18.5.4. Подключение браузера файлов к дереву (Демо-проект)

В разделе описано, как отобразить содержимое файловой системы в виде древовидной структуры. Для этого представлен проект-пример (файлы проекта идут в комплекте с документацией), демонстрирующий совместную работу компонента **Окружение: браузер файлов** со вспомогательными компонентами **Окружение: ошибка работы с файлами** и **Окружение: файлы**. В проекте-примере `Demo.hmi` будет проверяться содержимое папки проекта `FileSystemBrowser.`

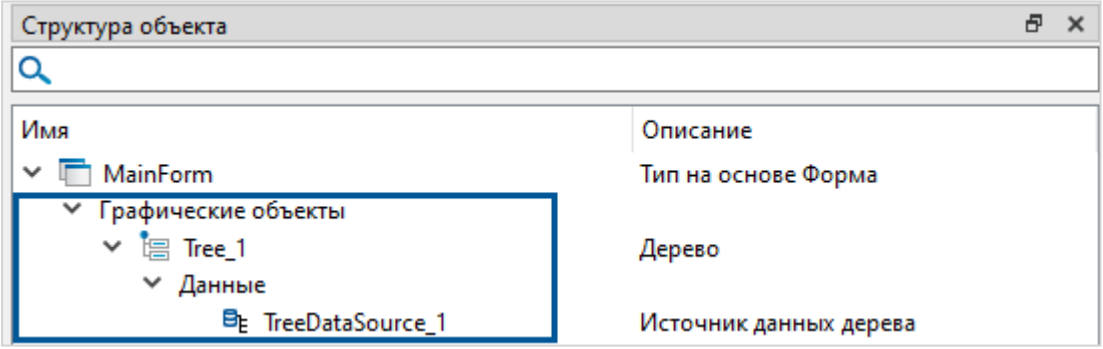
1. Добавьте на экранную форму объект типа **Дерево**.

2. В **Структуре объектов** щелкните правой кнопкой мыши по компоненту **Дерево** → **Создать**. Откроется мастер создания элементов.

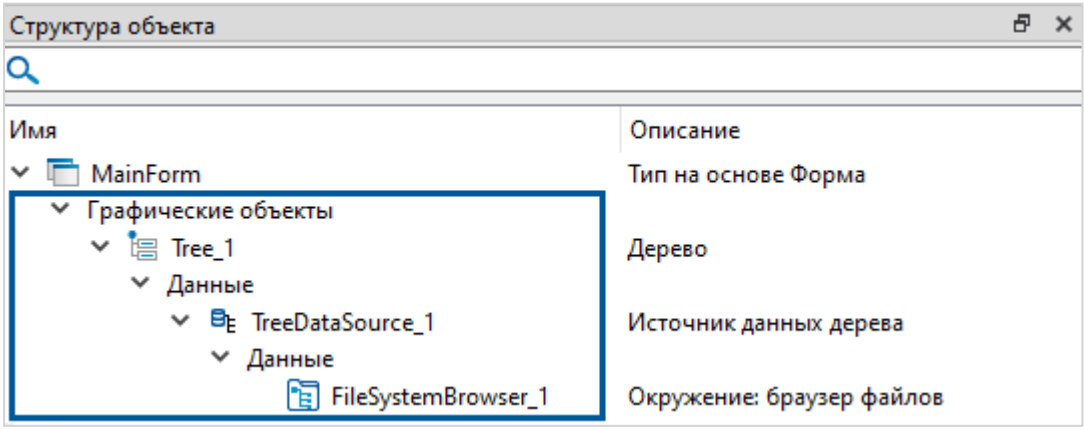
2.1. В поле **Тип элемента** оставьте **Экземпляр типа** , **Базовый тип элемента** → **Источник данных дерева**(раздел Визуальные) → **Создать**.



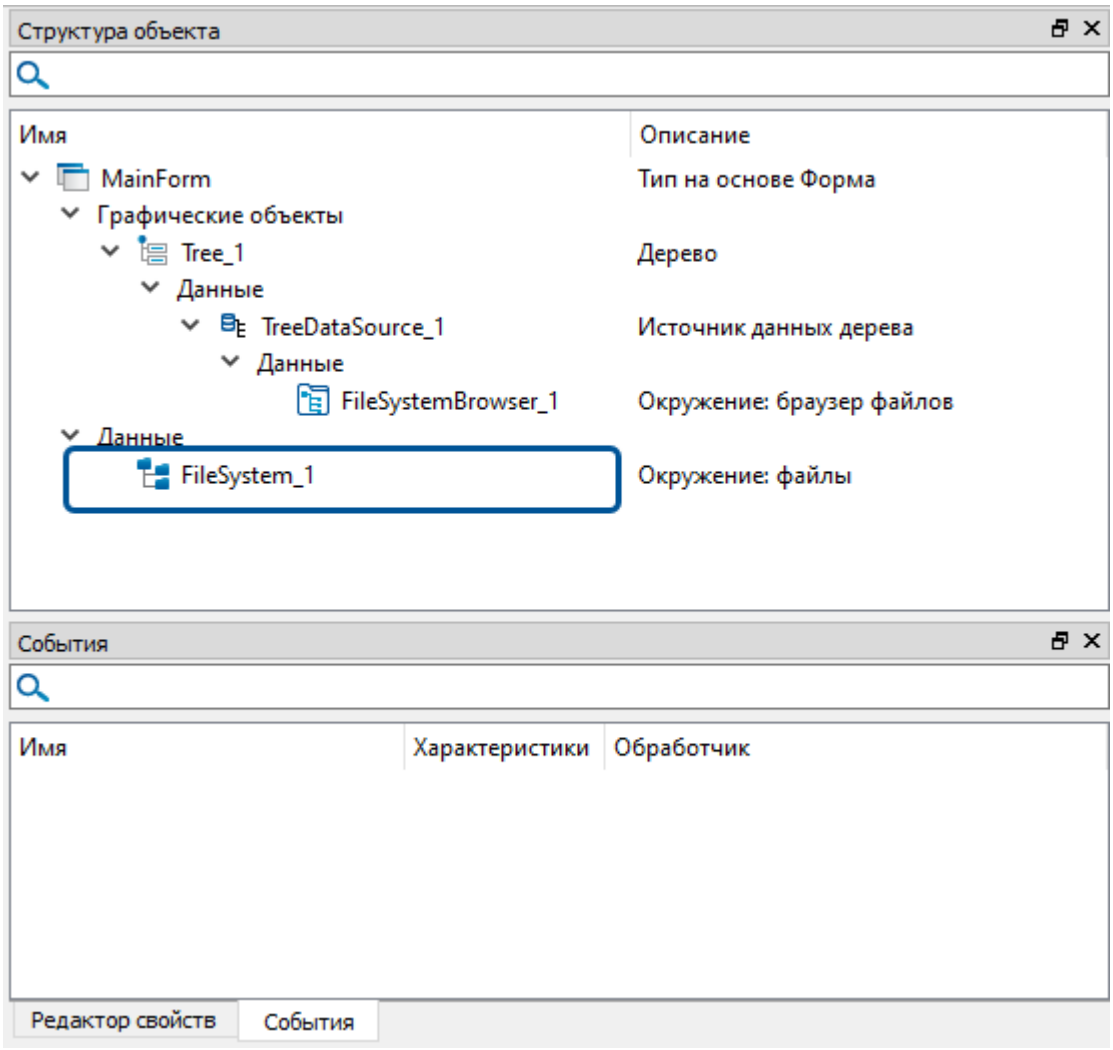
2.2. В **Структуре объектов** добавится компонент **Источник данных дерева** дочерним компоненту **Дерево**.



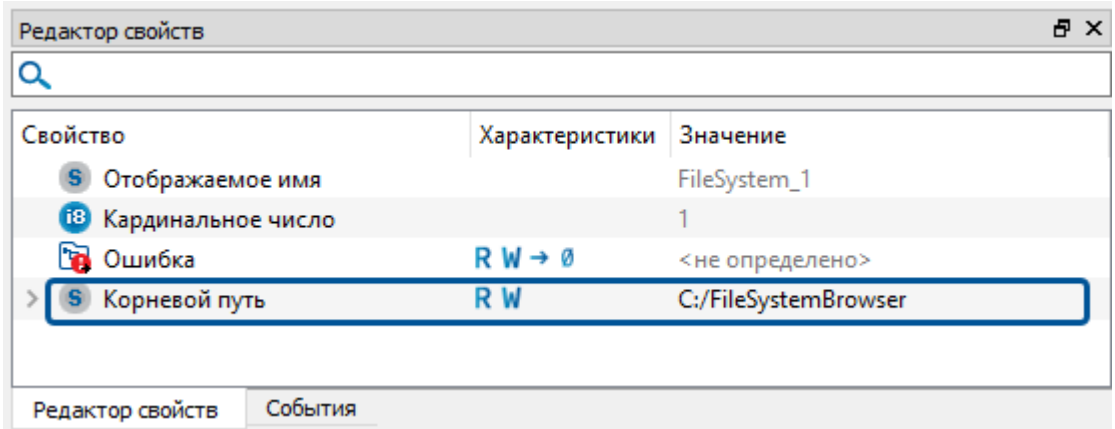
3. Для объекта **Источник данных дерева** аналогичным образом добавьте **Окружение: браузер файлов**.



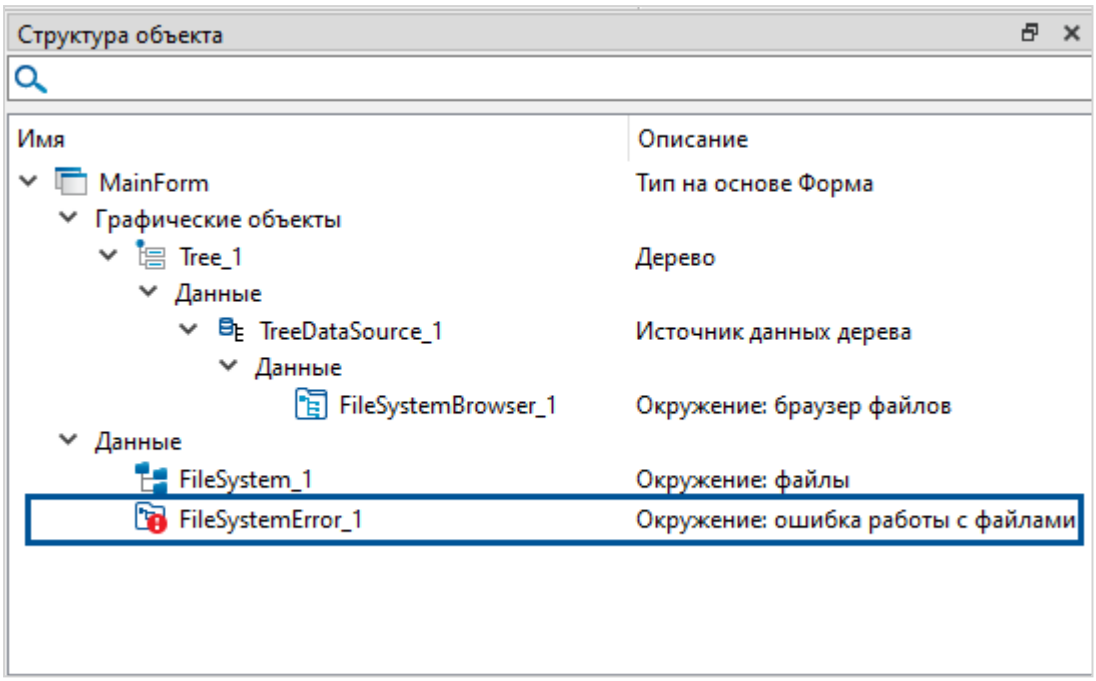
4. Чтобы работать с каталогами файловой системы, читать и записывать содержимое файлов, добавьте на экранную форму компонент **Окружение: файлы**.



4.1. Укажите корневой путь, с которого начинать просмотр файловой системы. Используйте для этого свойство **Корневой путь** (стр. 1) у объекта **Окружение: файлы**.

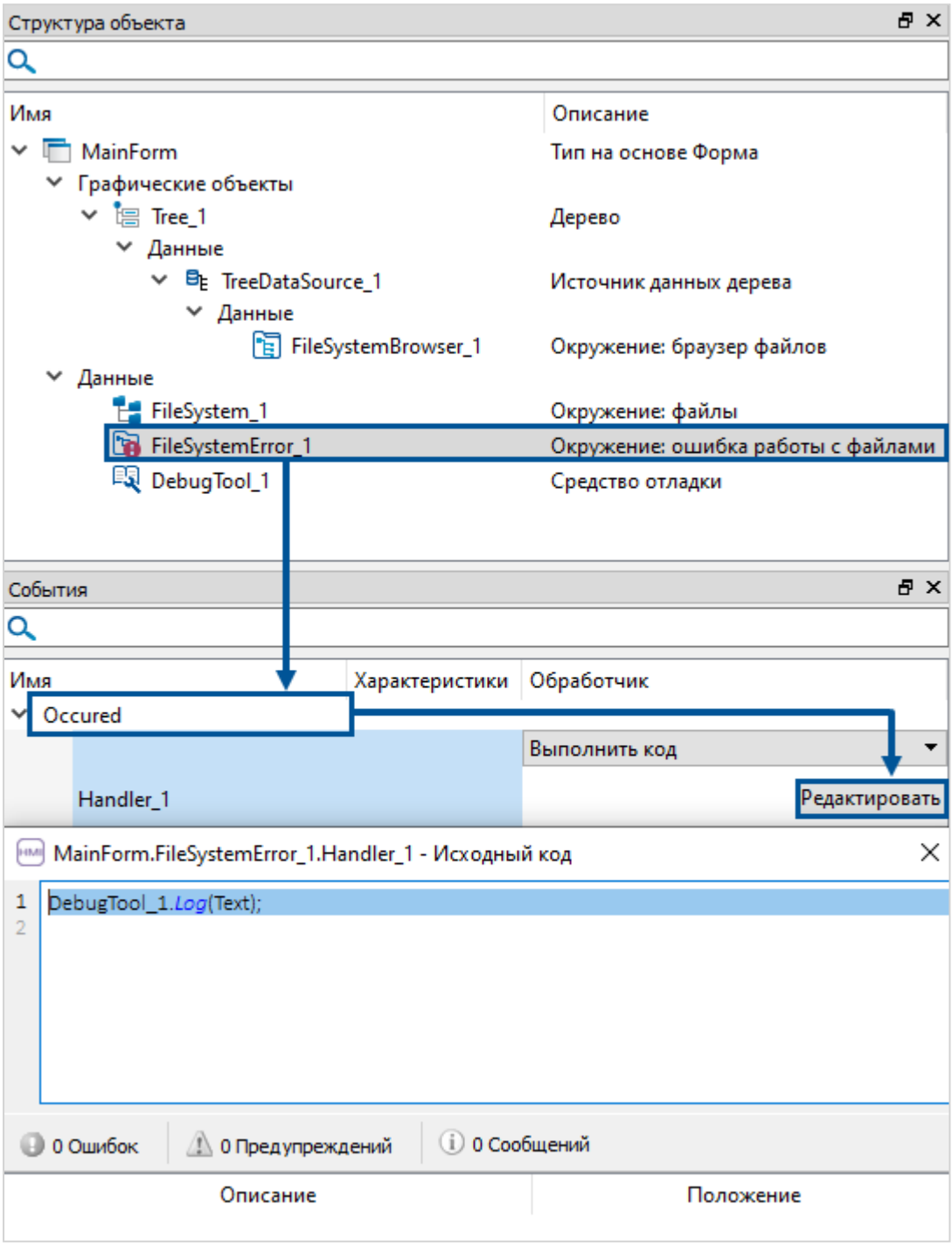


5. Чтобы иметь доступ к ошибкам, возникшим при выполнении какой-либо операции с файлом, добавьте компонент **Окружение: ошибка работы с файлами**.



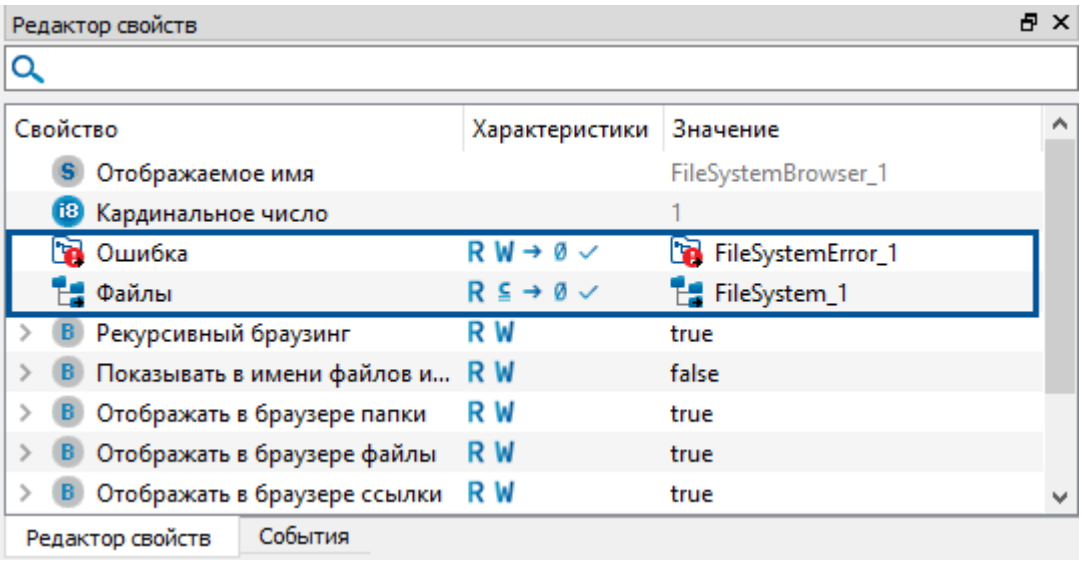
6. Чтобы выводить текст ошибки (свойство **Текст**) при выполнении последней операции в **Журнал времени исполнения** , добавьте на экранную форму компонент **Средство отладки**.

6.1. Выделите компонент **Окружение: ошибка работы с файлами**, перейдите во вкладку **События**, нажмите на событие **Occured** правым кликом мыши→ **Добавить обработчик** → **Выполнить код** → **Редактировать**. В открывшемся окне введите следующий код:

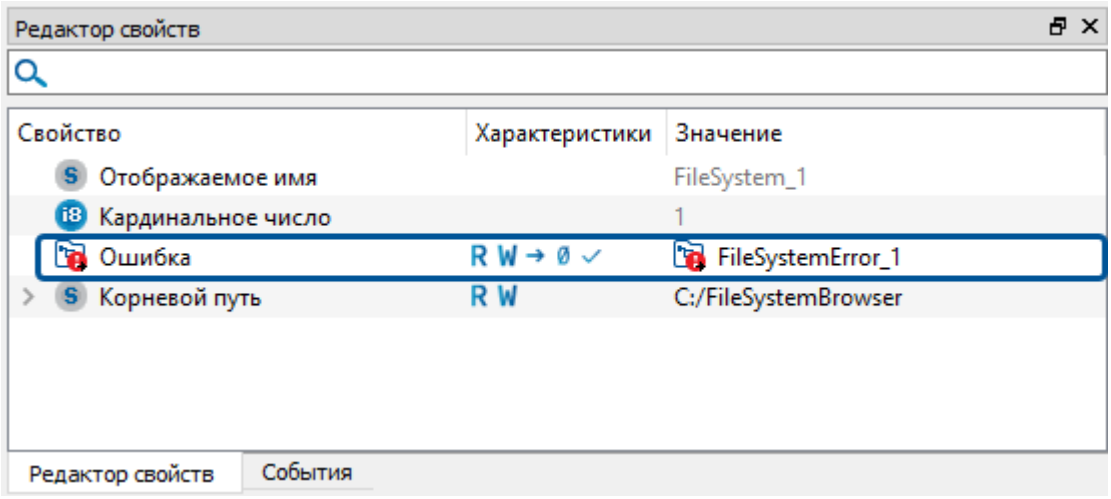



```
DebugTool_1.Log(Text);
```

7. У объекта **Окружение: браузер файлов** для свойств Ошибка и Файлы укажите в значении ссылку на добавленные ранее объекты типа **Окружение: ошибка работы с файлами** и **Окружение: файлы**. Для это нажмите правой кнопкой мыши по свойству Ошибка → **Сослаться** → **FileSystemError_1**. Аналогичным образом для свойства Файлы укажите ссылку на объект **FileSystem_1**.



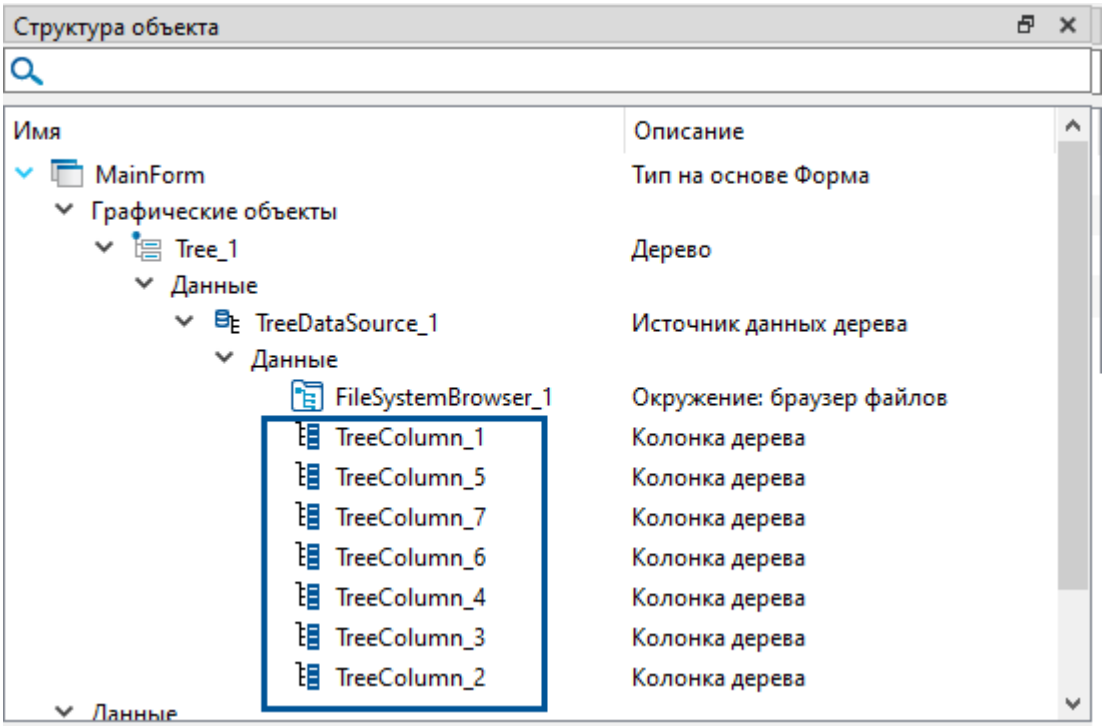
7.1. У объекта **Окружение: файлы** для свойства Ошибка также укажите в значении ссылку на добавленный объект типа **Окружение: ошибка работы с файлами**.

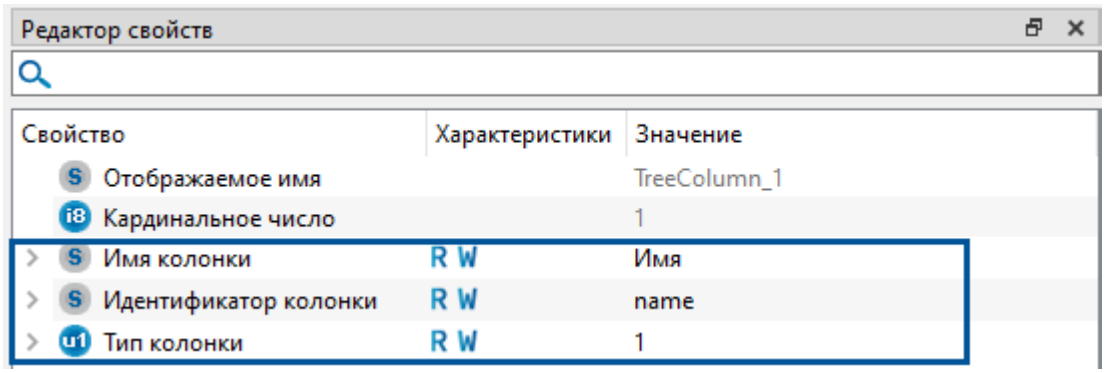


8. В **Источник данных дерева** добавьте **Колонки дерева** с идентификаторами:
   - **«name»**
   - **«id»**
   - **«parent_id»**
   - **«ext»**
   - **«size»**
   - **«created»**
   - **«expandable»**

   8.1. Используйте значения Идентификатора колонки и Тип колонки в соответствии с таблицей возвращаемых значений в разделе Какие данные возвращает компонент (стр. 213), которая поможет правильно настроить свойства для каждой **Колонки дерева**. Имя колонки указывается в произвольном виде.

> ПРИМЕЧАНИЕ
>
> **Колонки дерева** с идентификаторами **«name»**, **«id»** и **«parent_id»** обязательны для добавления на форму, все остальные добавляются при необходимости.

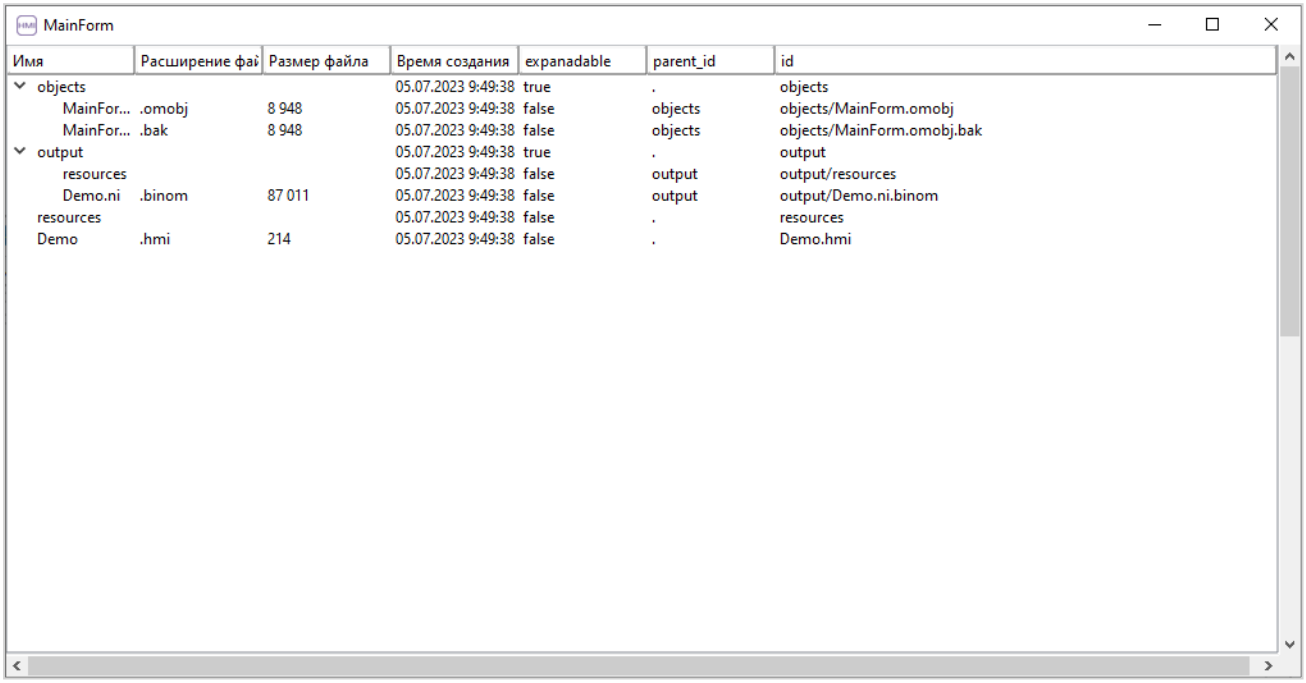




9. Начало новой сессии просмотра файловой системы инициируется с помощью функции **Browse**. Для того чтобы приступить к браузингу сразу после открытия экранной формы в рантайме, воспользуйтесь обработчиком события **Opened** и пропишите в нем код:

```
Tree_1.TreeDataSource_1.FileSystemBrowser_1.Browse(""); // !NB: Браузинг начнется с
корневого пути, который был задан на шаге 4.1
```

10. Запустите форму в рантайме, чтобы посмотреть содержимое папки проекта `FileSystemBrowser`.



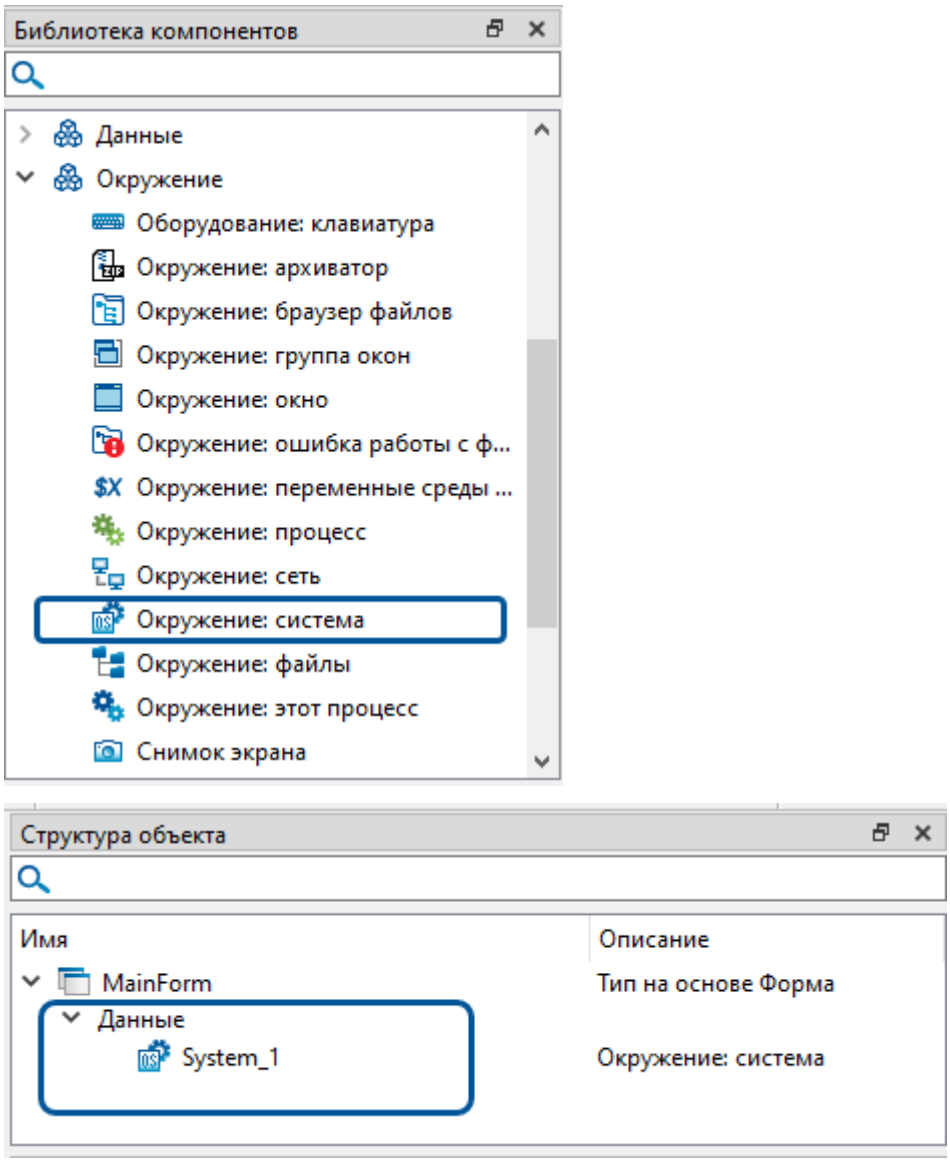
18.6. Получение информации об операционной системе и среде исполнения SePlatform.HMI

Для получения информации о системе, на которой запущено приложение SePlatform.HMI, а также о среде, в которой это приложение функционирует, используйте компоненты: **Окружение: система** и **Среда исполнения**.

18.6.1. Окружение: система

Компонент **Окружение: система** предоставляет информацию о текущей операционной системе, на которой работает приложение SePlatform.HMI. Компонент позволяет получить различные характеристики системы, такие как название операционной системы, версия, семейство, ядро, версия ядра и порядок байтов.

Компонент **Окружение: система** расположен в юните **«Окружение»** библиотеки компонентов. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.

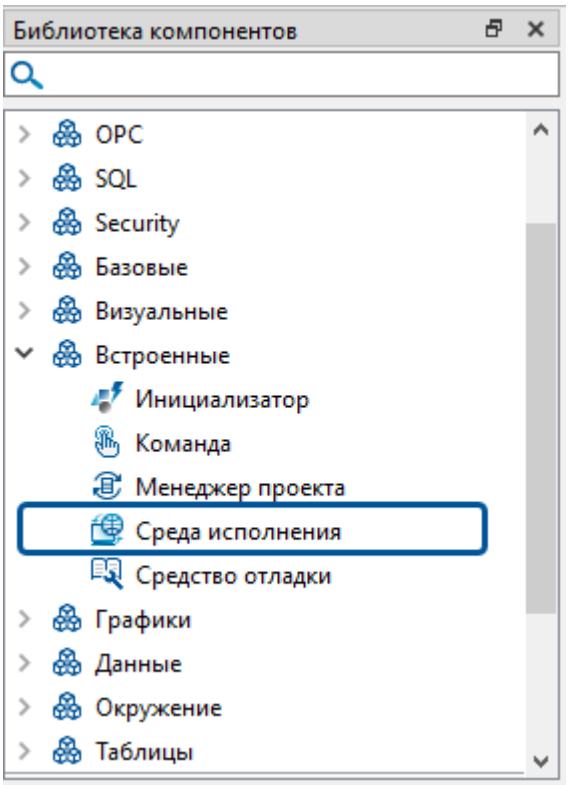


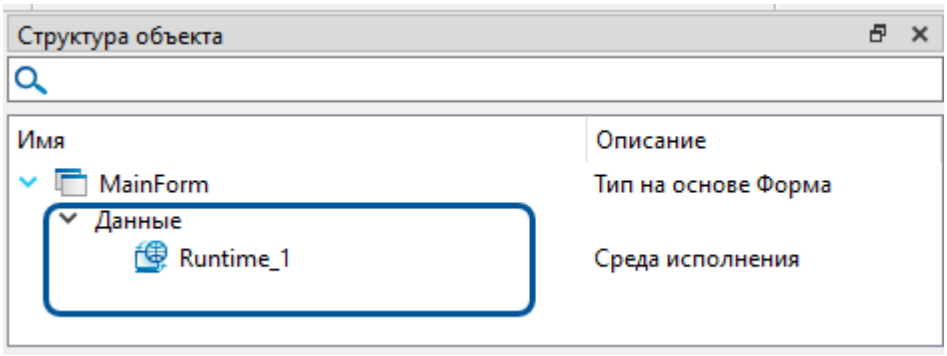
Свойства компонента **Окружение: система** рассмотрены в справочном руководстве.

18.6.2. Среда исполнения

Компонент **Среда исполнения** предоставляет информацию о текущей среде выполнения приложения SePlatform.HMI. Компонент позволяет определить режим работы приложения: **«desktop»** или **«web»**.

Компонент **Среда исполнения** расположен в юните **«Встроенные» библиотеки компонентов**. Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.





Свойства компонента **Среда исполнения** рассмотрены в справочном руководстве.

18.6.3. Пример использования компонентов

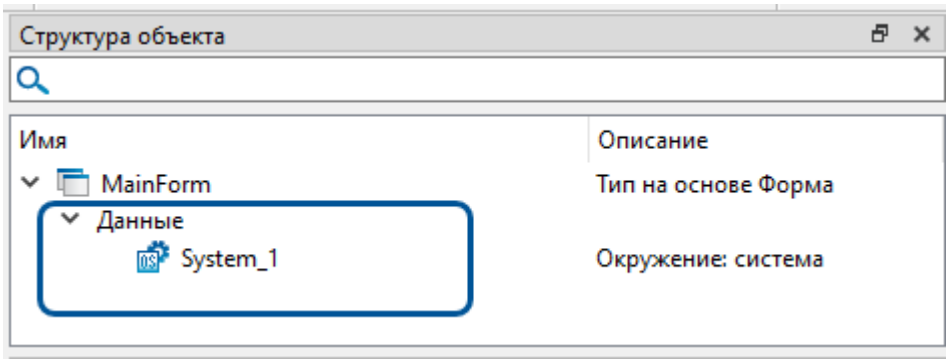
Далее кратко описан пример (скачать его можно здесь) использования компонентов **Окружение: система** и **Среда исполнения** в проекте SePlatform.HMI с использованием языка SePlatform.Om.

В примере показано, как:

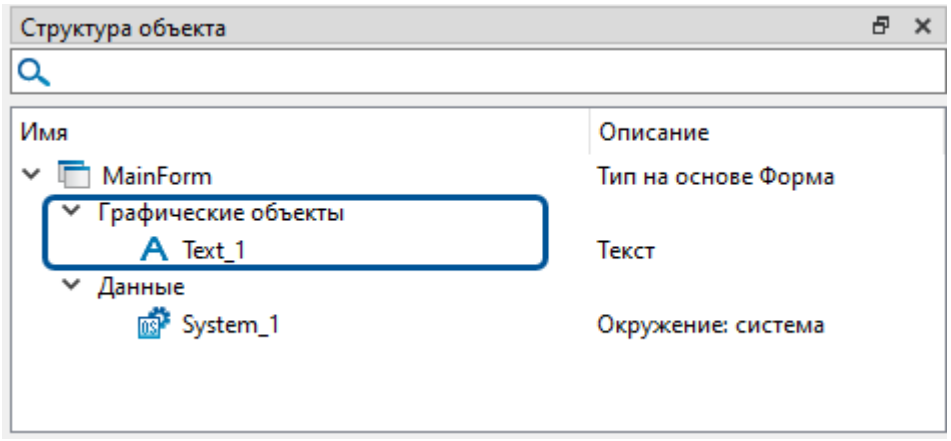
- Получить информацию о текущей операционной системе, на которой работает приложение SePlatform.HMI, с помощью компонента **Окружение: система**.
- Определить режим работы приложения (**«desktop»** или **«web»**) с использованием компонента **Среда исполнения**.

18.6.3.1. Получить информацию о текущей операционной системе

1. Для получения информации о текущей операционной системе добавьте на экранную форму компонент **Окружение: система**.



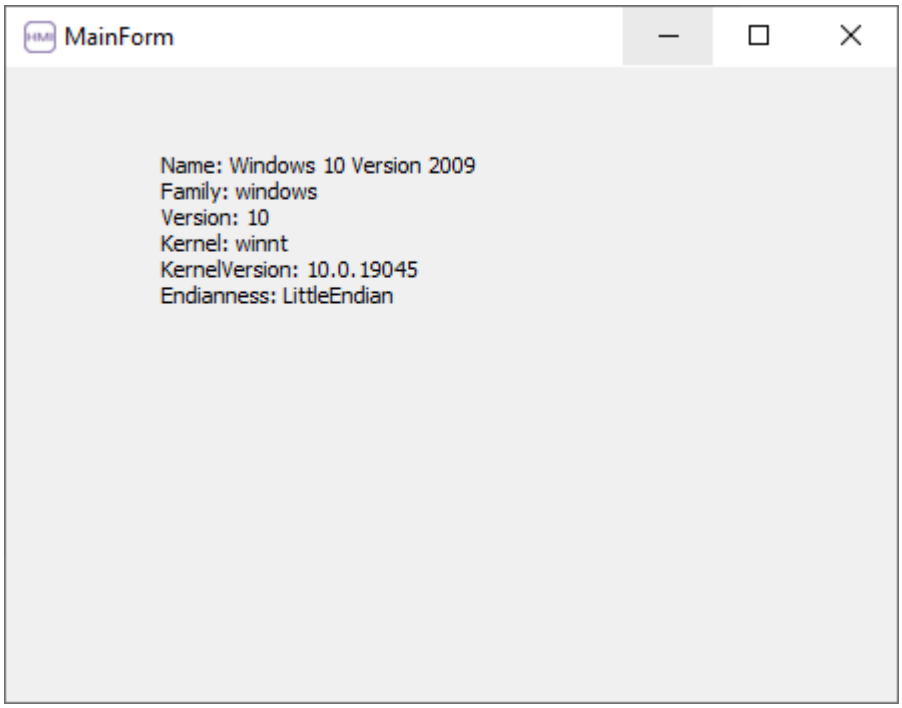
2. Подготовьте текстовое поле, в котором будет отображаться информация о системе.



3. Воспользуйтесь обработчиком события **Opened** для просмотра информации о системе сразу после открытия экранной формы. В этом обработчике пропишите следующий код:

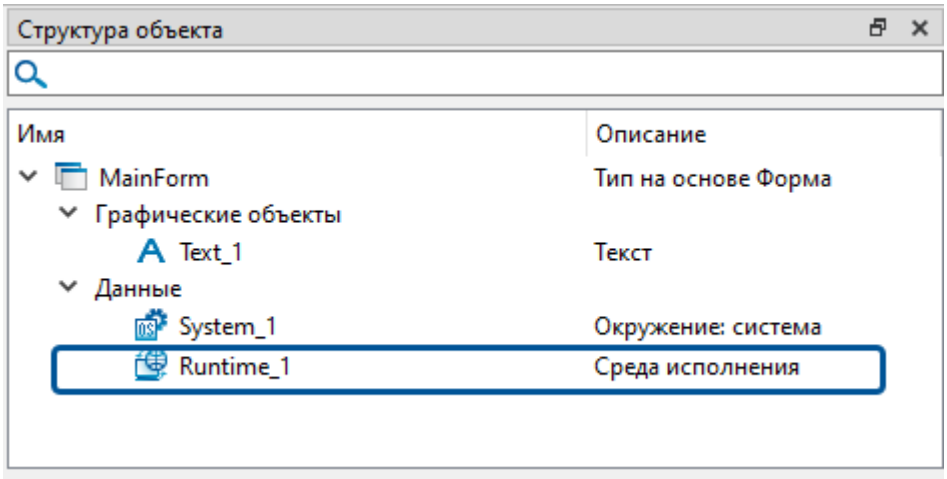
```
// Получаем информацию о системе и выводим её в текстовое поле
Text_1.Text = "Name: " + System_1.Name + "\n" +
    "Family: " + System_1.Family + "\n" +
    "Version: " + System_1.Version + "\n" +
    "Kernel: " + System_1.Kernel + "\n" +
    "KernelVersion: " + System_1.KernelVersion + "\n" +
    "Endianness: " + (System_1.Endianness == 0 ? "BigEndian" : "LittleEndian");
```

4. Запустите форму в режиме рантайм, и информация о системе будет отображена в текстовом поле.

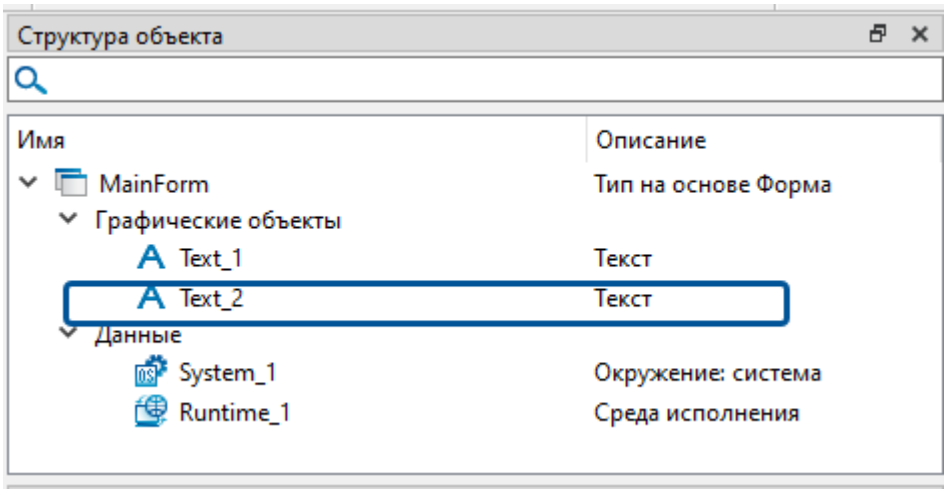


18.6.3.2. Определить режим работы приложения

1. Для получения информации о текущей среде выполнения приложения SePlatform.HMI добавьте на экранную форму компонент **Среда исполнения**.



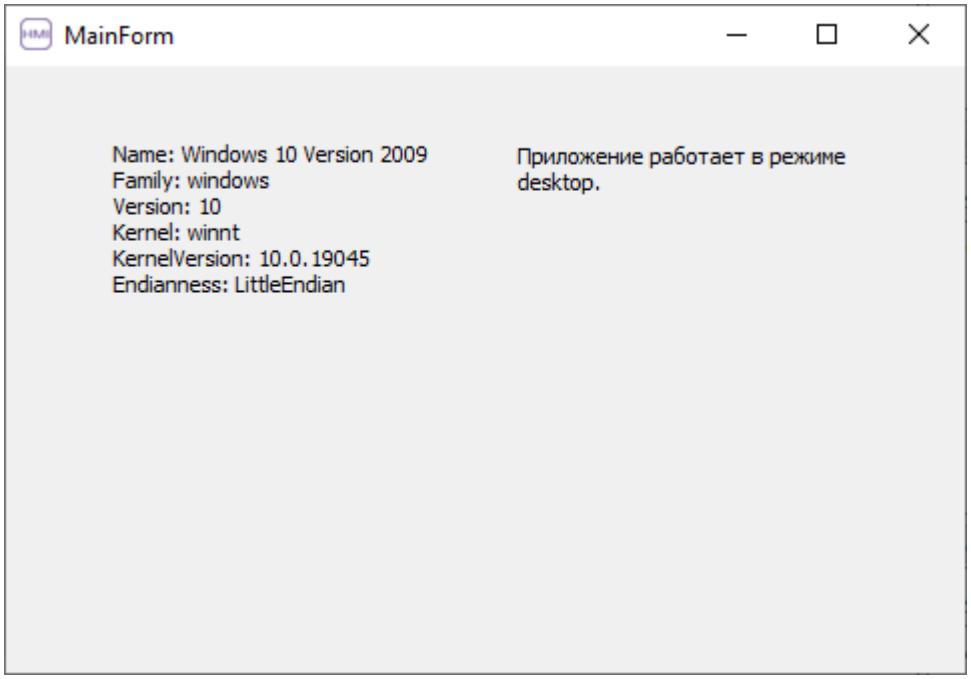
2. Подготовьте дополнительное текстовое поле, в котором будет отображаться информация о режиме работы приложения.



3. Воспользуйтесь обработчиком события **Opened** для получения информации о режиме работы приложения сразу после открытия экранной формы. В этом обработчике пропишите следующий код:

```
// Объявление переменной для хранения информации о режиме работы приложения
currentMode: uint1 = Runtime_1.WorkMode;
modeInfo: string;
if ( currentMode == 0) {
      modeInfo = "Приложение работает в режиме desktop.";
 } else if (currentMode == 1) {
      modeInfo = "Приложение работает в режиме web.";
 }
Text_2.Text = modeInfo; // Выводим информацию о режиме работы приложения в текстовое
поле
```

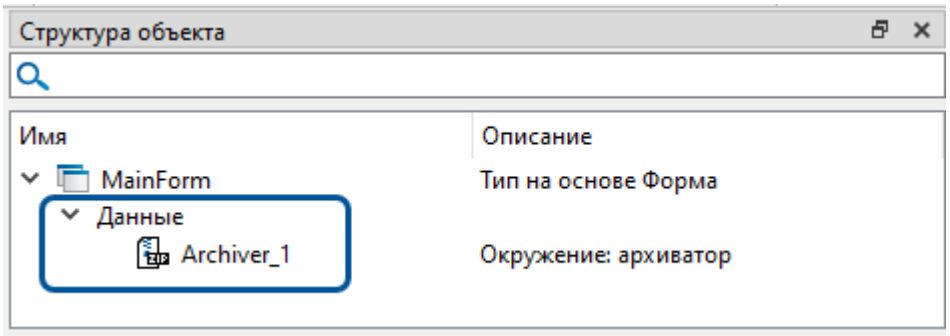
4. Запустите форму в режиме рантайм, и информация о режиме работы приложения будет отображена в текстовом поле.



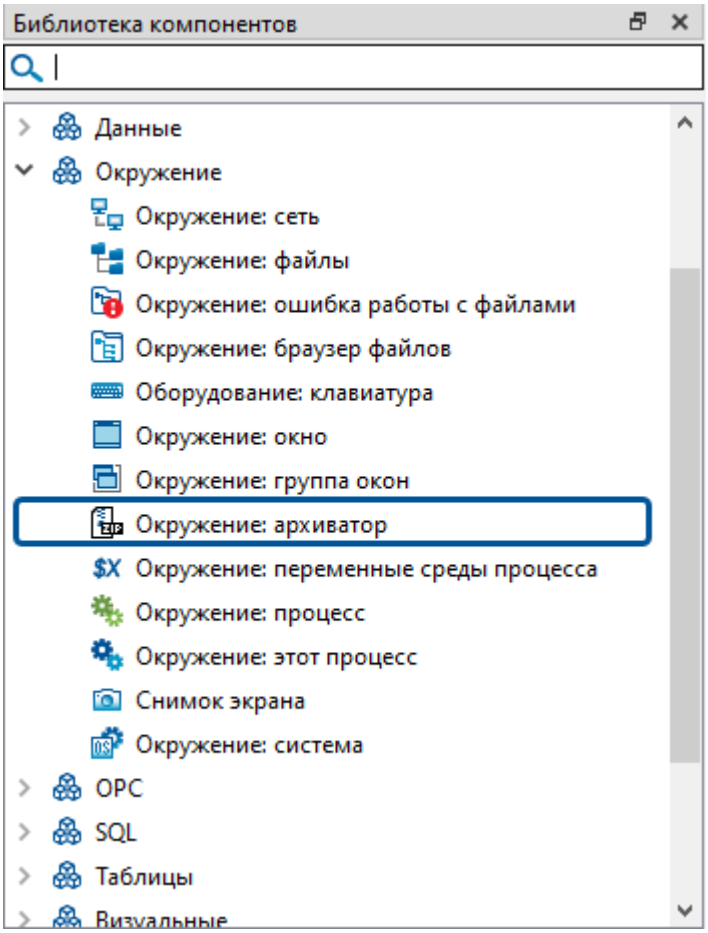
18.7. Архивирование файлов

Чтобы архивировать и распаковывать файлы, воспользуйтесь компонентом **Окружение: архиватор**. Компонент поддерживает основные форматы архивов, включая `zip`, `7z`, `rar` и другие.

Компонент **Окружение: архиватор** расположен в юните **«Окружение» библиотеки компонентов**.



Экземпляр этого типа не отображается на форме и виден только в области **Структура объекта**.



18.8. Демо-пример работы с архивами

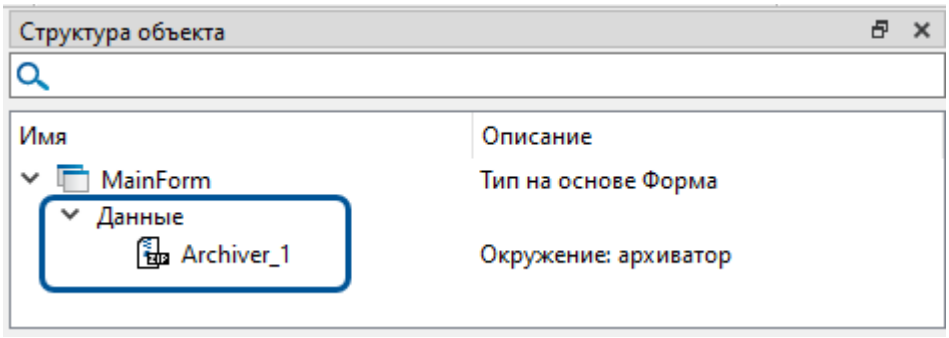
Далее кратко описан небольшой пример (файлы проекта идут в комплекте с документацией) использования компонента **Окружение: архиватор** в проекте SePlatform.HMI с использованием языка SePlatform.Om.

В примере показано как:

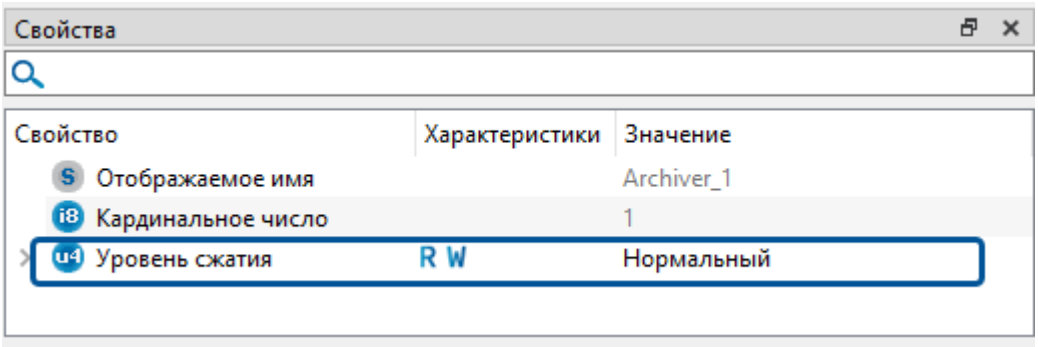
- Упаковать файлы в формат `zip` и сохранить архив в выбранную папку;
- Распаковать архив в выбранную папку;
- Посмотреть список файлов и папок в архиве без распаковки.

18.8.1. Упаковать файлы в формат zip и сохранить архив в выбранную папку

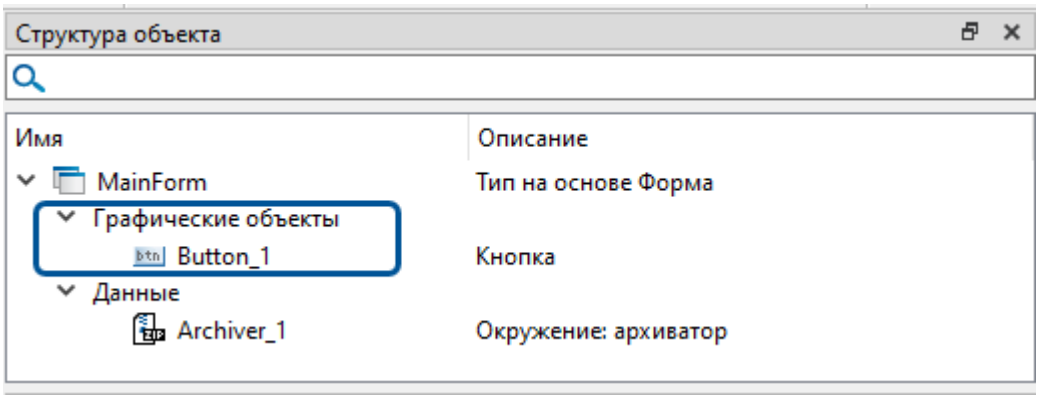
1. Добавьте на форму компонент **Окружение: архиватор**.



2. Выберите в свойстве Уровень сжатия желаемую степень сжатия.



3. Добавьте на форму кнопку, при нажатии на которую будет производиться упаковка файлов в формат `zip` и сохраняться по указанному пути.

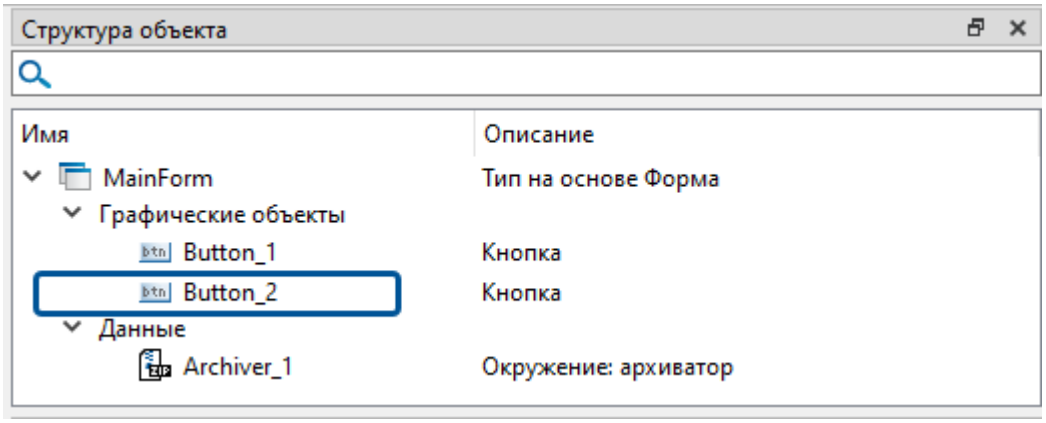


4. Настройте обработчик события ButtonPressed для кнопки. В обработчике используйте метод AsyncPack компонента **Окружение: архиватор**, чтобы упаковать файлы в архив и сохранить архив в выбранную папку. Пример кода:

```
Archiver_1.AsyncPack("C:/DEMO_PROJECT", "C:/Archive/DEMO_PROJECT.zip");
```

18.8.2. Распаковать архив в выбранную папку

1. Добавьте на форму кнопку, при нажатии на которую будет производиться распаковка архива формата `zip` в выбранную папку.

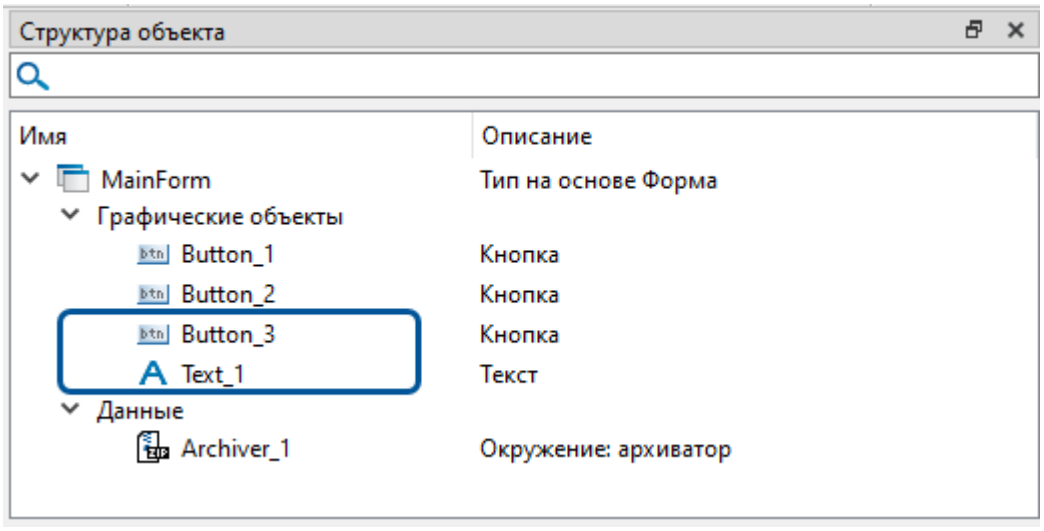


2. Настройте обработчик события ButtonPressed для кнопки. В обработчике используйте метод AsyncUnpack компонента **Окружение: архиватор**, чтобы распаковать архив в выбранную папку. Пример кода:

```
Archiver_1.AsyncUnpack("C:/Archive/DEMO_PROJECT.zip", "C:/DEMO_PROJECT");
```

18.8.3. Посмотреть список файлов и папок в архиве без распаковки

1. Добавьте на форму кнопку и подготовьте текстовое поле, куда будет выводится список файлов из архива.



2. Настройте обработчик события ButtonPressed для кнопки. В обработчике используйте метод BrowseAsJSON компонента **Окружение: архиватор**, чтобы получить список файлов и папок в архиве без распаковки. Пример кода:

```
Text_1 = Archiver_1.BrowseAsJSON("C:/Archive/DEMO_PROJECT.zip");
```

19. Печать графических элементов

Для печати графических элементов в проектах SePlatform.HMI используются следующие компоненты:

- Компонент **Печать** позволяет отправлять на печать выбранные графические элементы.
- Компонент **Системные принтеры** обеспечивает доступ к детальной информации о всех принтерах подключенных к системе. Компонент предоставляет такие данные, как названия принтеров, их описания, поддерживаемые разрешения печати и другие параметры.
- Компонент **Принтер** предназначен для настройки параметров печати на конкретном принтере. Вы можете настроить такие параметры, как режим цветности печати, количество копий, ориентацию страниц и другие опции. Используйте этот компонент в сочетании с компонентом **Печать** для отправки на печать.

Для получения подробной информации о настройке данных компонентов, а также примерах их использования, обратитесь к демонстрационному примеру ниже.

19.1. Как напечатать графический элемент

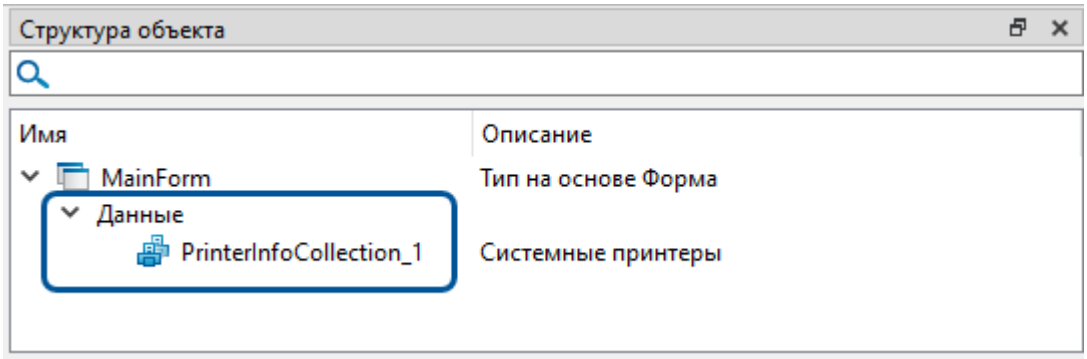
Далее кратко описан небольшой пример использования компонентов печати в проекте SePlatform.HMI с использованием языка SePlatform.Om.

В примере показано, как:

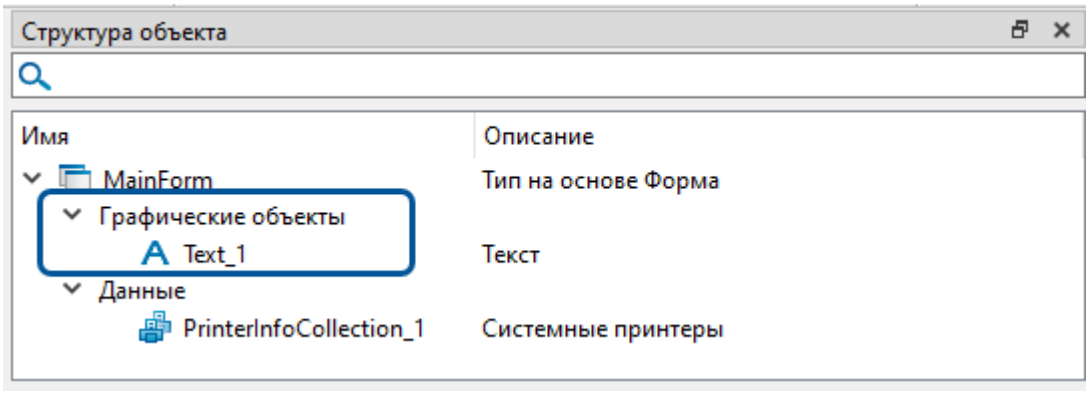
- Получить список всех принтеров и определить название принтера, установленного по умолчанию;
- Настроить параметры печати с помощью компонента **Принтер**;
- Отправить на печать графический элемент через компонент **Печать**.

19.1.1. Получить список всех принтеров и определить название принтера, установленного по умолчанию

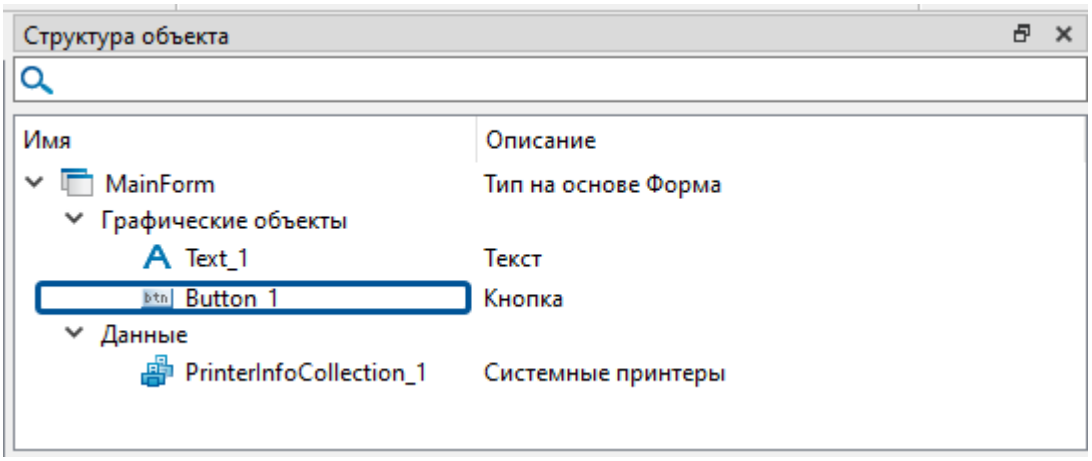
1. Разместите на форме компонент **Системные принтеры**, который будет использован для доступа к информации о принтерах.



2. Добавьте текстовое поле, которое будет отображать список всех принтеров и имя принтера, используемое по умолчанию.



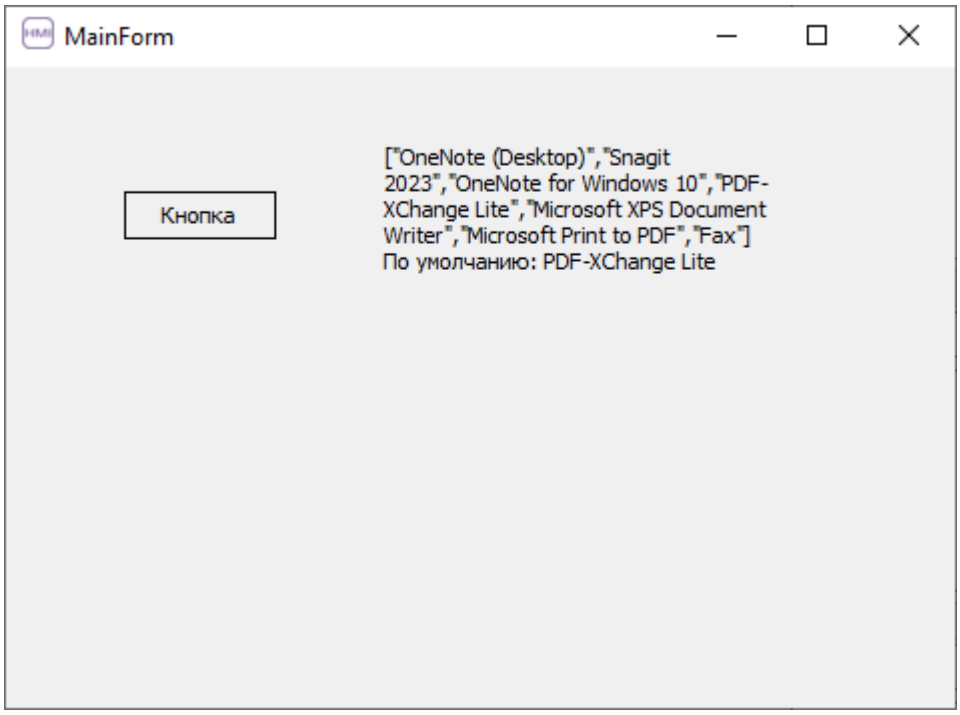
3. Поместите кнопку на форме, нажатие на которую инициирует запрос на получение списка принтеров и определение принтера по умолчанию.



4. Настройте обработчик события ButtonPressed для кнопки, чтобы при её нажатии активировался процесс получения и отображения информации о принтерах. Пример кода:

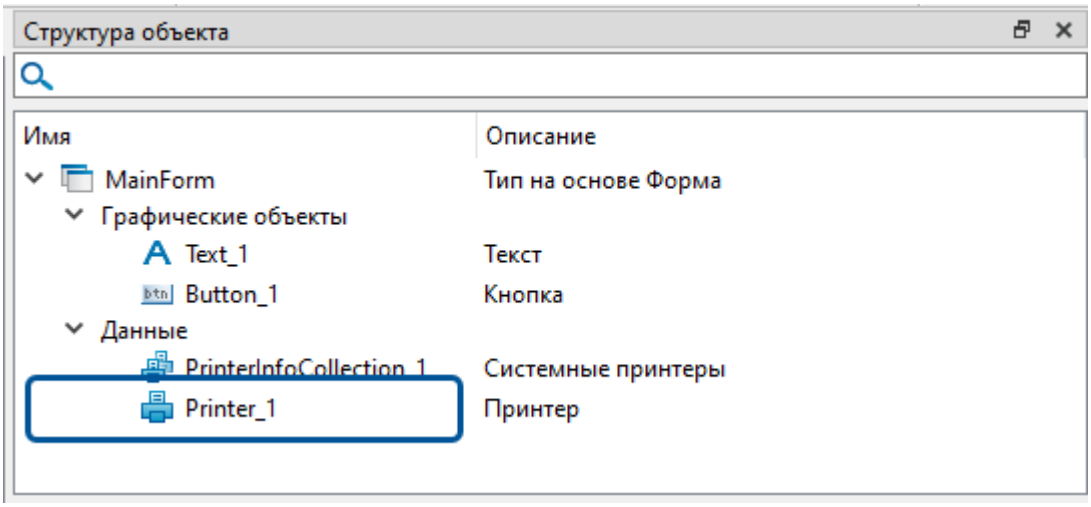
```
// Устанавливаем текст в текстовом поле: список всех принтеров и принтер по умолчанию
Text_1.Text = PrinterInfoCollection_1.PrinterNames + "\nПо умолчанию: " +
PrinterInfoCollection_1.PrinterDefaultName;
```

5. Запустите форму в режиме рантайм, и после нажатия на кнопку **«Button_1»** в текстовое поле будет выведен список доступных принтеров и название принтера, установленного в системе по умолчанию.

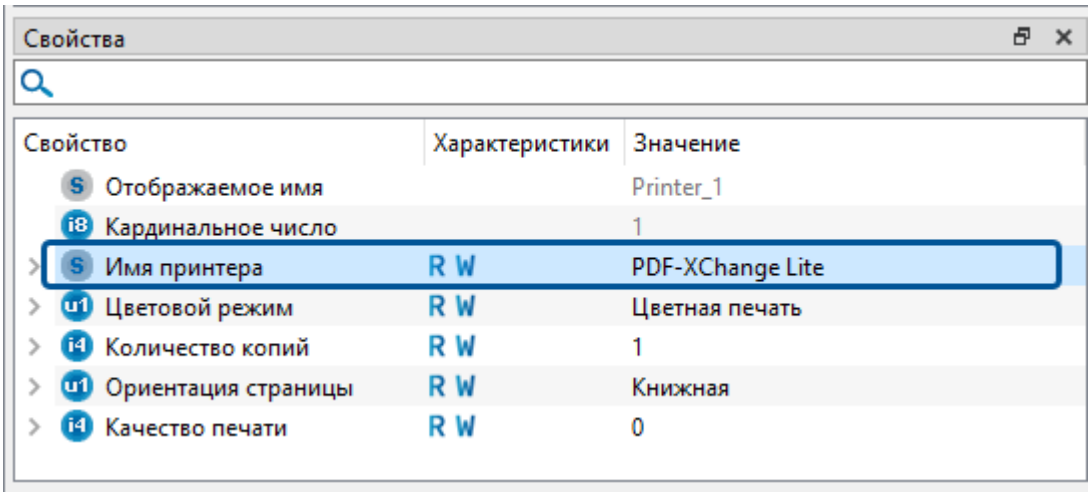


19.1.2. Настроить параметры печати

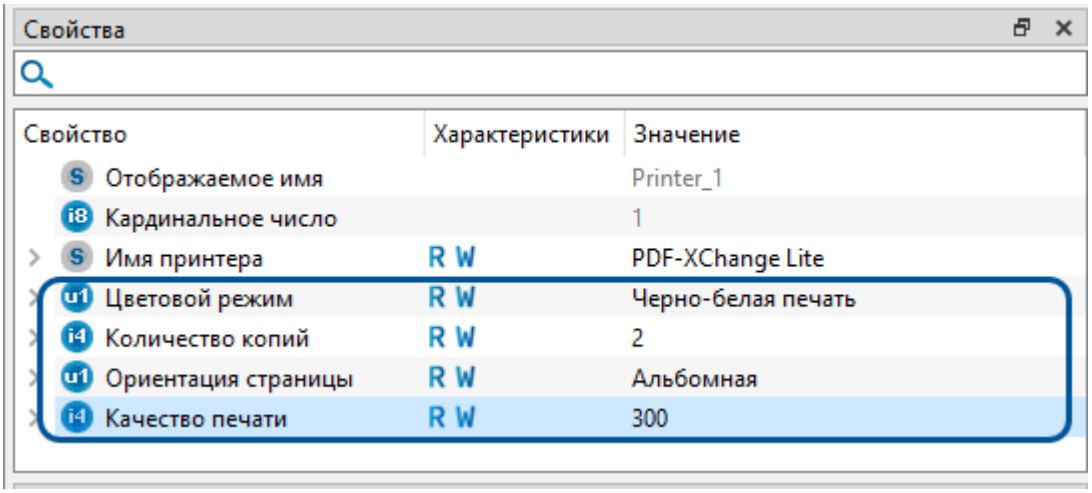
1. Добавьте на форму компонент **Принтер**, который будет использоваться для настройки печати.



2. В свойстве Имя принтера компонента **Принтер** установите принтер, определенный в предыдущем разделе как используемый по умолчанию, или выберите другой из списка.

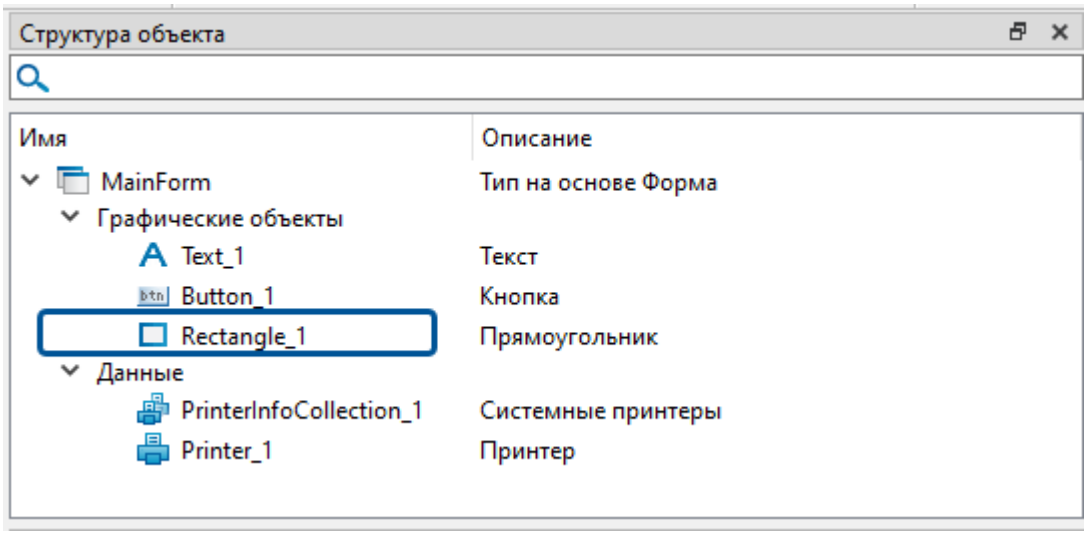


3. Настройте другие параметры печати, такие как Цветовой режим, Количество копий, Ориентация страницы и Качество печати , используя соответствующие свойства.

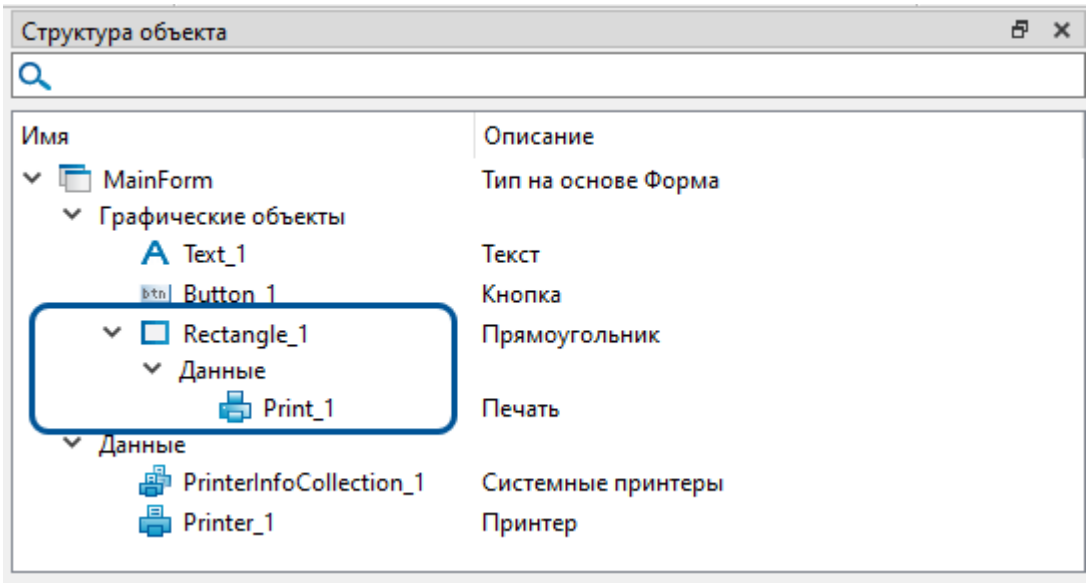


19.1.3. Отправить на печать графический элемент

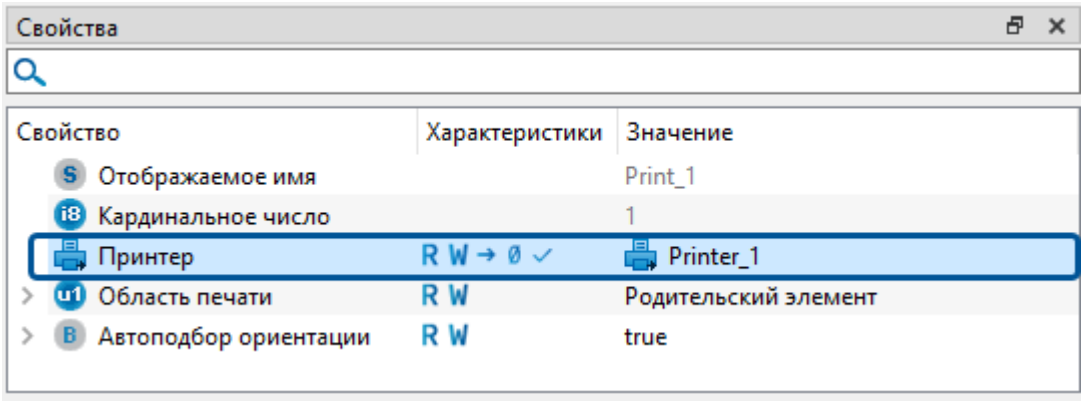
1. Добавьте на форму графический элемент, который вы хотите напечатать.



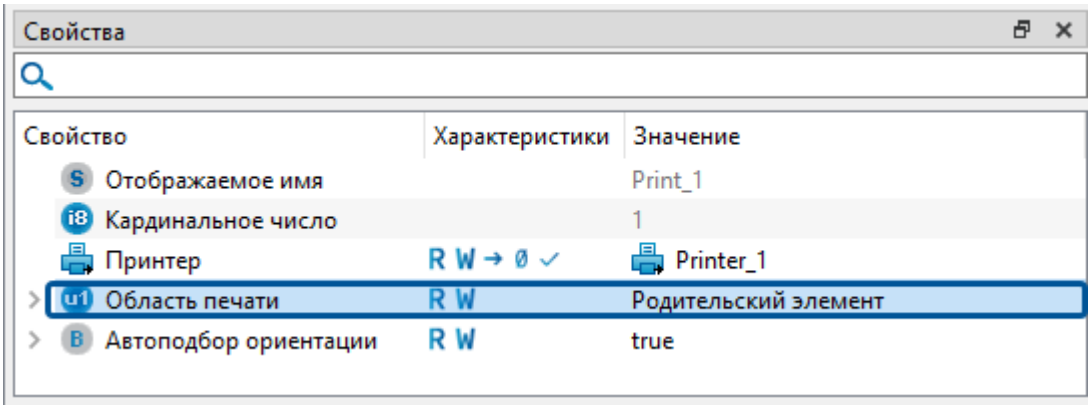
2. Добавьте компонент **Печать** в качестве дочернего к графическому элементу.



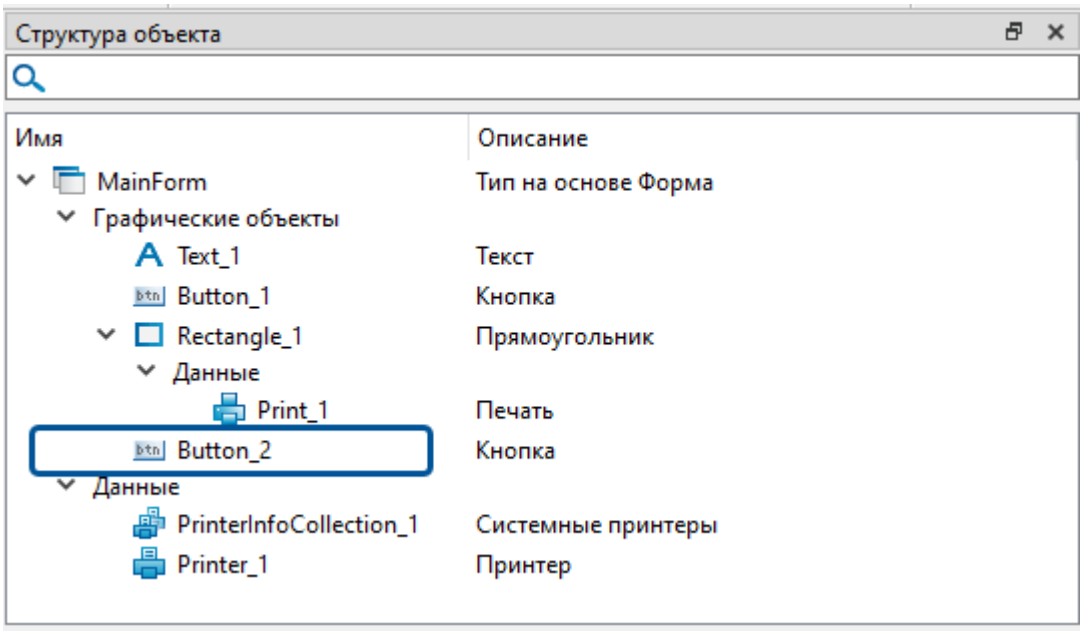
3. Укажите ссылку на добавленный ранее компонент **Принтер** для компонента **Печать** через свойство Принтер. Для этого нажмите правой кнопкой мыши по свойству Принтер→ **Сослаться** →**Printer_1**.



4. Настройте в компоненте **Печать** свойство Область печати, выбирая, что именно будет напечатано. Для печати самого графического элемента выберите **«Родительский элемент»**.



5. Разместите кнопку на форме, при нажатии на которую будет запускаться процесс печати.



6. Настройте обработчик события **ButtonPressed** для кнопки, чтобы при её нажатии запускался процесс печати. Пример кода:

```
Rectangle_1.Print_1.Print(); // Команда для запуска печати графического элемента
```

7. Запустите форму в режиме рантайм, и после нажатия на кнопку **«Button_2»** будет активирован процесс печати выбранного графического элемента.

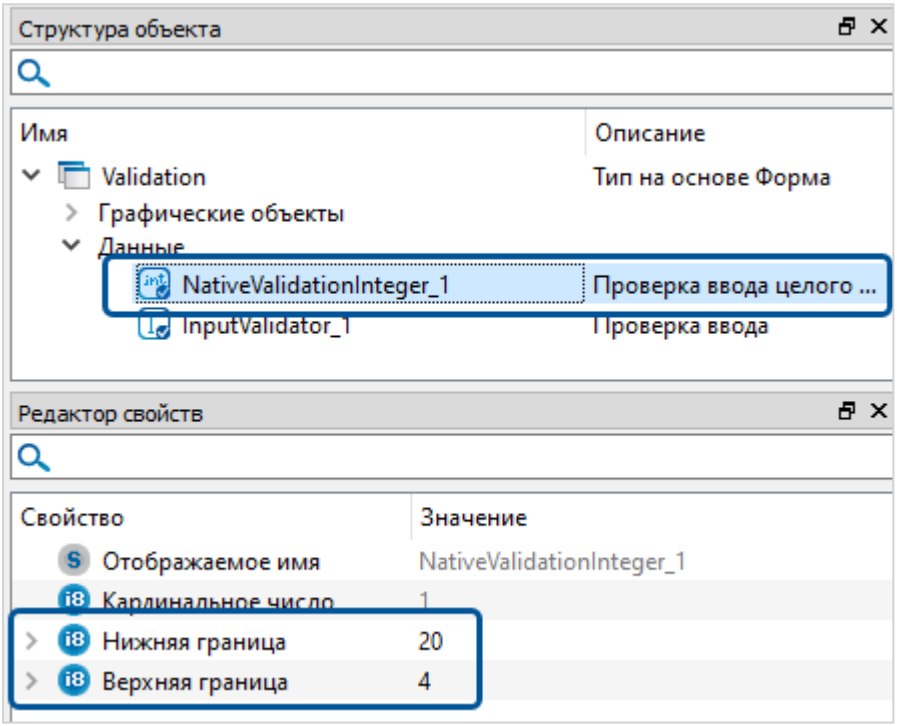
20. Валидация данных

SePlatform.HMI предоставляет набор компонентов для проверки различных типов данных на соответствие заданным условиям и ограничениям.

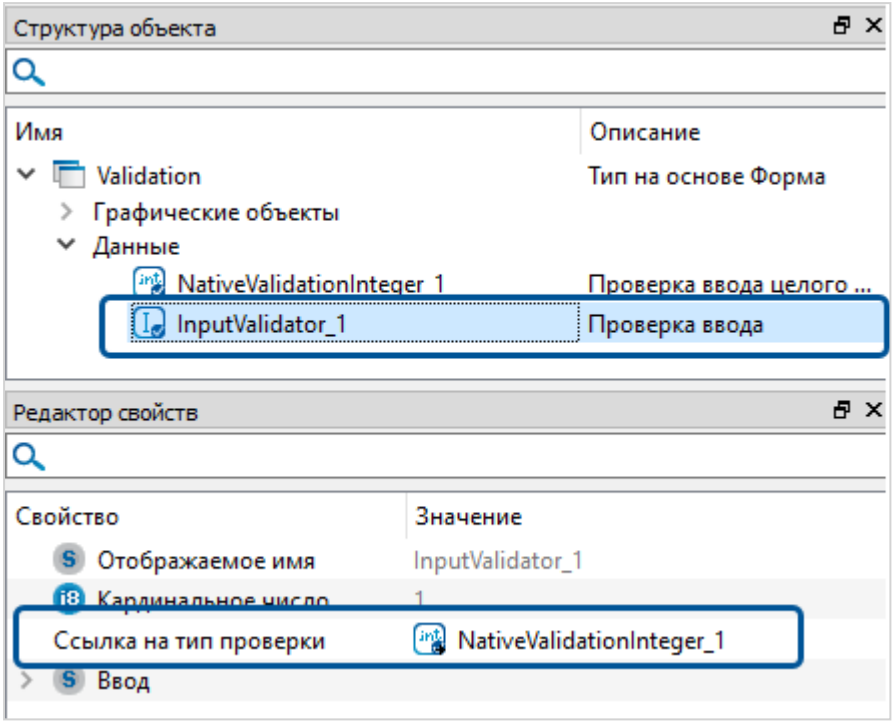
20.1. Валидация целых чисел

Для проверки целочисленных значений на соответствие заданным условиям выполните следующие действия:

1. Определите условия валидации: добавьте на рабочую область компонент библиотеки **Проверка ввода целого числа** (из группы **`om.automation.controls`**) и укажите значения свойств Нижняя граница и Верхняя граница.



2. Добавьте на рабочую область компонент библиотеки **Проверка ввода** и сошлитесь в свойстве Ссылка на тип проверки на условие валидации, созданное в предыдущем шаге.



3. Запишите значение, которое следует валидировать, в свойство Input:

```
InputValidator.Input = "15";
```

4. Получите результат валидации из свойства Result:

```
Text_1.Text = Str.ToString(InputValidator.Result);
```

Свойство Result может принимать следующие условные коды:

- **«0»** - значение не валидировано;
- **«1»**- значение валидировано частично;
- **«2»** - значение валидировано полностью.

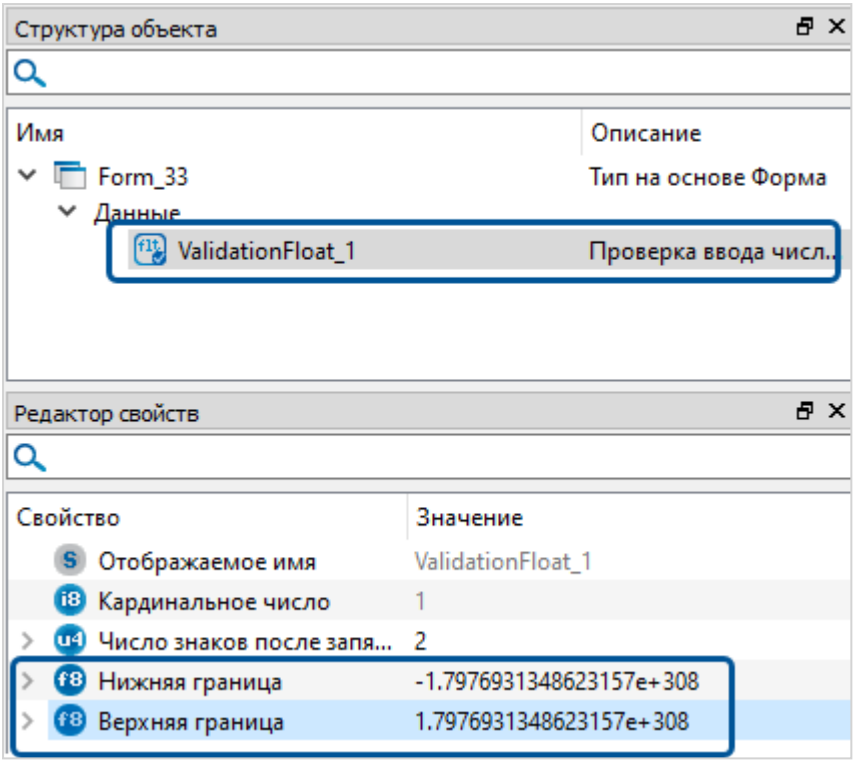
ОБРАТИТЕ ВНИМАНИЕ

Если требуется реакция на каждую смену результата валидации, то воспользуйтесь специальным событием ResultChanged.

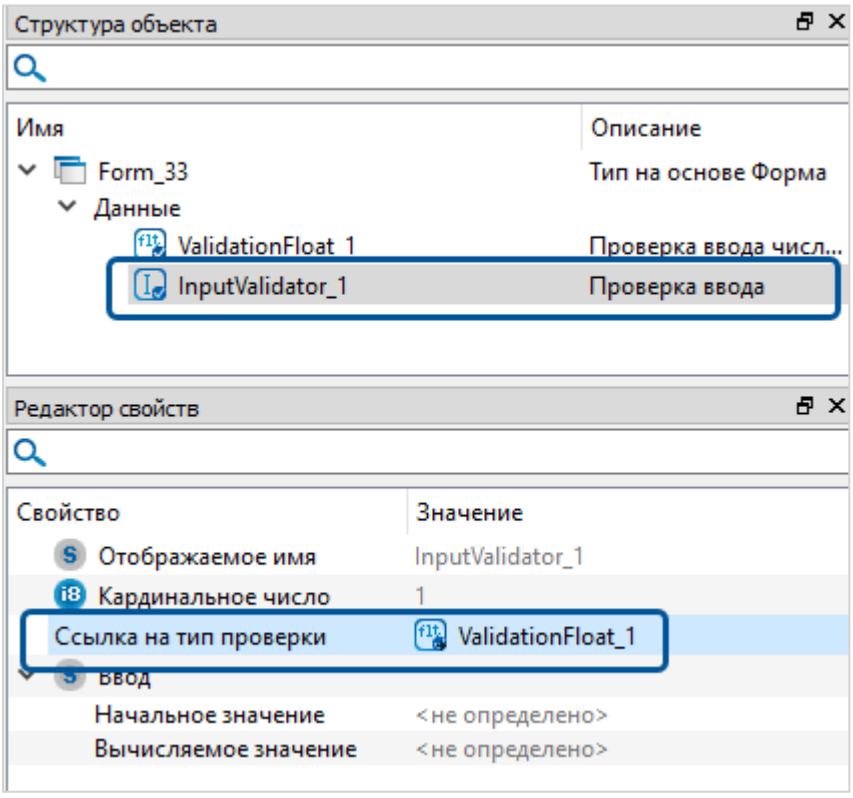
20.2. Валидация чисел с плавающей запятой

Для проверки чисел с плавающей запятой на соответствие заданным условиям выполните следующие действия:

1. Определите условия валидации: добавьте на рабочую область компонент библиотеки **Проверка ввода числа с плавающей запятой** (из группы **om.automation.controls**) и укажите значения свойств Число знаков после запятой, Нижняя граница, Верхняя граница.



2. Добавьте на рабочую область компонент библиотеки **Проверка ввода** и сошлитесь в свойстве **Ссылка на тип проверки** на условие валидации, созданное в предыдущем шаге.



3. Запишите значение, которое следует валидировать, в свойство **Input**:

```
InputValidator.Input = "1.33";
```

4. Получите результат валидации из свойства **Result**:

```
Text_1.Text = Str.ToString(InputValidator.Result);
```

Свойство **Result** может принимать следующие условные коды:

- **«0»** - значение не валидировано;
- **«1»** - значение валидировано частично;
- **«2»** - значение валидировано полностью.

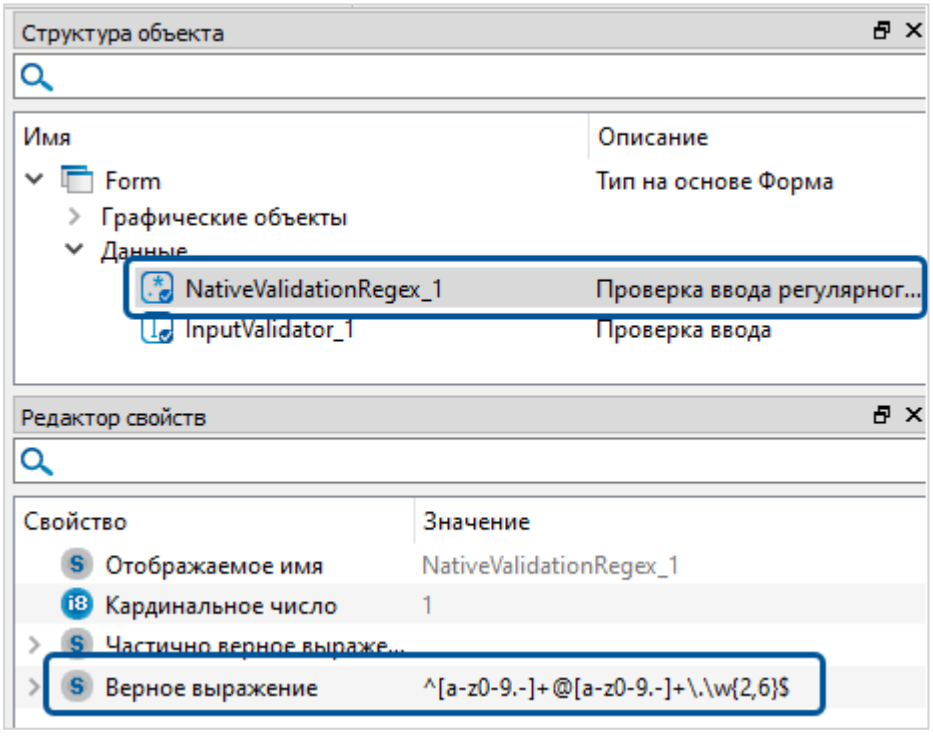
ОБРАТИТЕ ВНИМАНИЕ

Если требуется реакция на каждую смену результата валидации, то воспользуйтесь специальным событием **ResultChanged**.

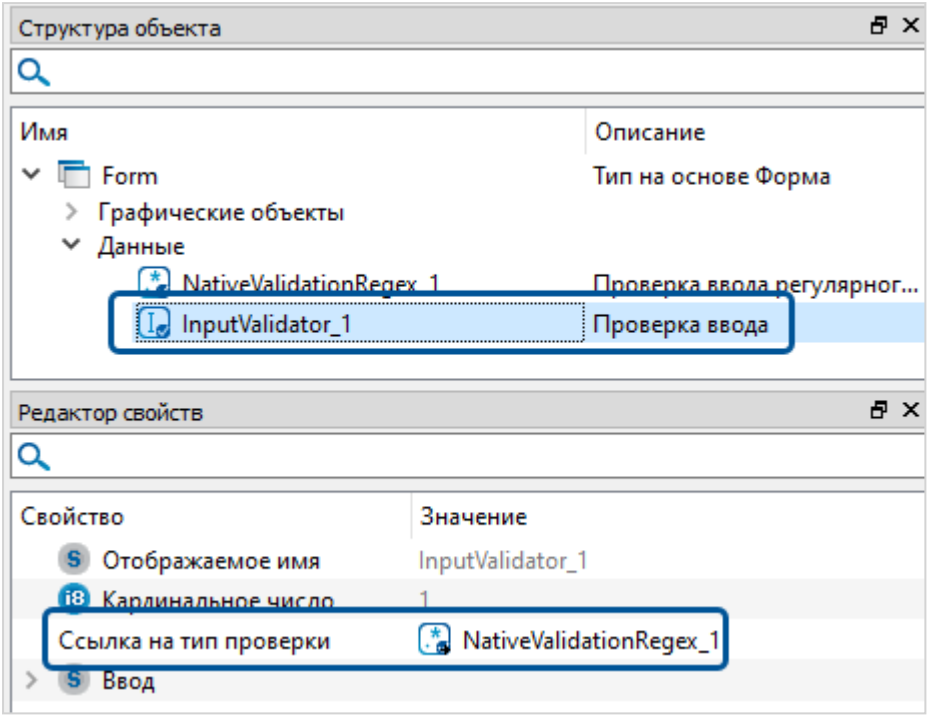
20.3. Валидация по регулярным выражениям

Для проверки строк на соответствие регулярному выражению выполните следующие действия:

1. Определите условия валидации: добавьте на рабочую область компонент библиотеки **Проверка ввода регулярного выражения** (из группы **om.automation.controls**) и укажите значения свойств **Частично верное выражение** и **Верное выражение**. На рисунке ниже показано регулярное выражение для проверки адреса электронной почты.



2. Добавьте на рабочую область компонент библиотеки **Проверка ввода** и сошлитесь в свойстве **Ссылка на тип проверки** на условие валидации, созданное в предыдущем шаге.



3. Запишите значение, которое следует валидировать, в свойство **Input**:

```
InputValidator.Input = "mymail@domain.com";
```

4. Получите результат валидации из свойства **Result**:

```
Text_1.Text = Str.ToString(InputValidator.Result);
```

Свойство **Result** может принимать следующие условные коды:

- **«0»** - значение не валидировано;
- **«1»** - значение валидировано частично;
- **«2»** - значение валидировано полностью.

ОБРАТИТЕ ВНИМАНИЕ

Если требуется реакция на каждую смену результата валидации, то воспользуйтесь специальным событием **ResultChanged**.

21. Преобразование форматов данных

SePlatform.HMI предоставляет набор компонентов для преобразования форматов данных.

21.1. Изменение формата

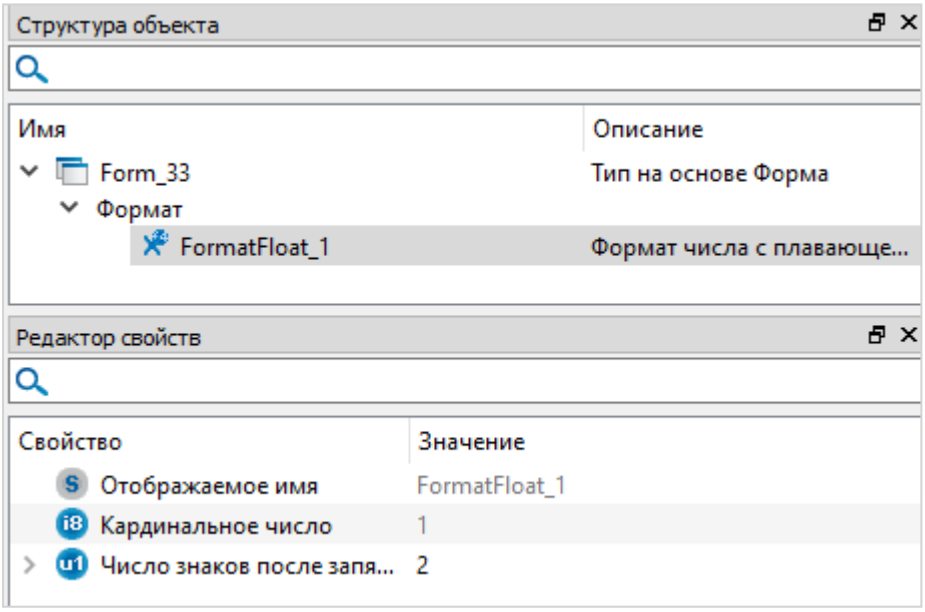
Компоненты **om.automation.controls** позволяют гибко представлять данные в форматах:

- число с плавающей точкой;
- булевское значение;
- дата и время.

Компоненты **om.automation.controls** работают в связке с компонентами **Входной формат** и **Выходной формат**.

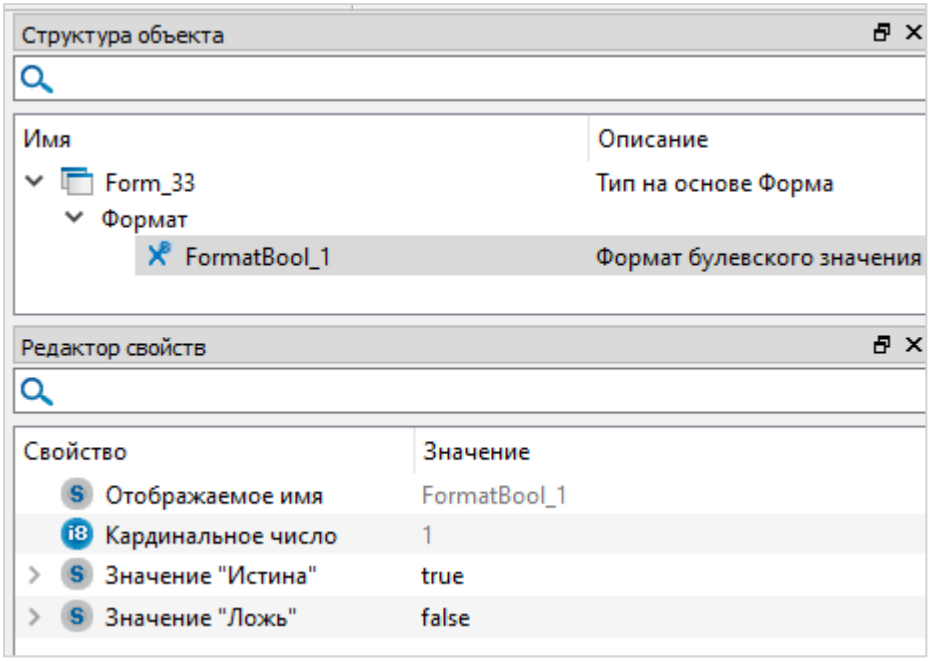
21.1.1. Формат числа с плавающей запятой

Компонент служит для представления чисел с плавающей точкой с нужной точностью. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на форму был добавлен компонент **Формат числа с плавающей запятой**. Единственным свойством компонента является Число знаков после запятой, с помощью которого определяется точность дробного числа.



21.1.2. Формат булевского значения

Компонент служит для представления логических значений («**true**»/«**false**») в альтернативном виде. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на форму был добавлен компонент **Формат булевского значения**.

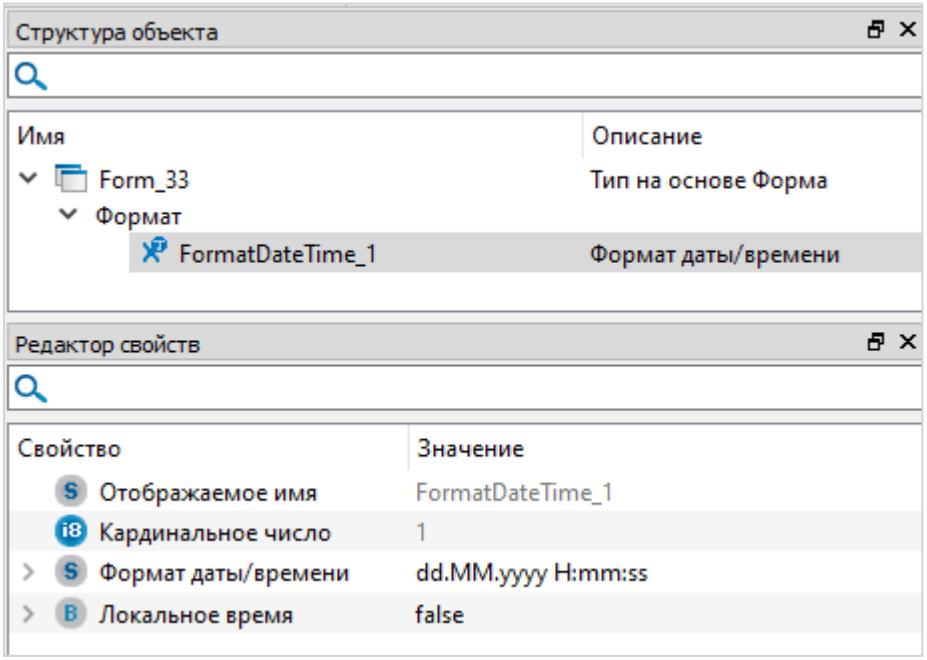


Чтобы определить альтернативное строковое представление для логических значений, установите значения свойств.

Свойство	Тип	Описание
Значение "Истина"	string	Строка, которая соответствует значению «**True**».
Значение "Ложь"	string	Строка, которая соответствует значению «**False**».

21.1.3. Формат даты и времени

Компонент служит для представления даты и времени в заданном формате. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на форму был добавлен компонент **Формат даты/времени**.



Для настройки формата отображения даты и времени установите свойства компонента:

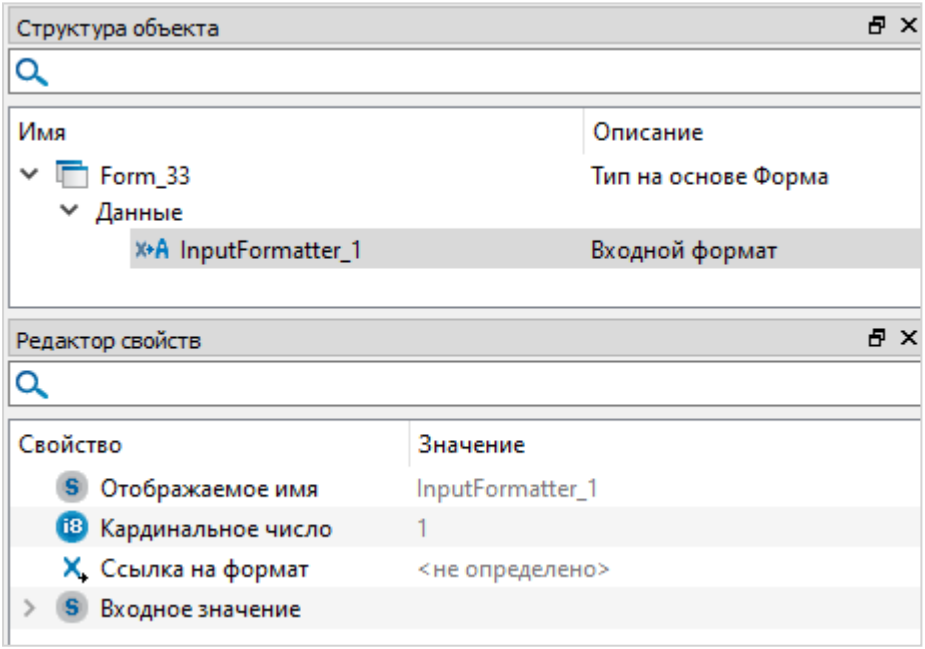
Свойство	Тип	Описание
Формат даты/времени	string	Формат представления метки времени. Доступны обозначения: «dd» - дни, «MM» - месяцы, «yyyy» - год, «H» - часы, «mm» - минуты, «ss» - секунды.
Локальное время	bool	Если свойство активно, то оригинальная метка времени заменяется локальным временем.

21.2. Конвертация типов данных

Для конвертации типов потребуется настроенный компонент типа **om.automation.controls**, который будет работать в связке с компонентами **Входной формат** или **Выходной формат**.

21.2.1. Конвертация строк в универсальный тип Variant

Чтобы конвертировать строки (значения типа string) в значения универсального типа variant, воспользуйтесь компонентом **Входной формат**. Добавьте на экранную форму компонент **Входной формат**. Компонент невизуальный и виден только в области **Структура объекта**.

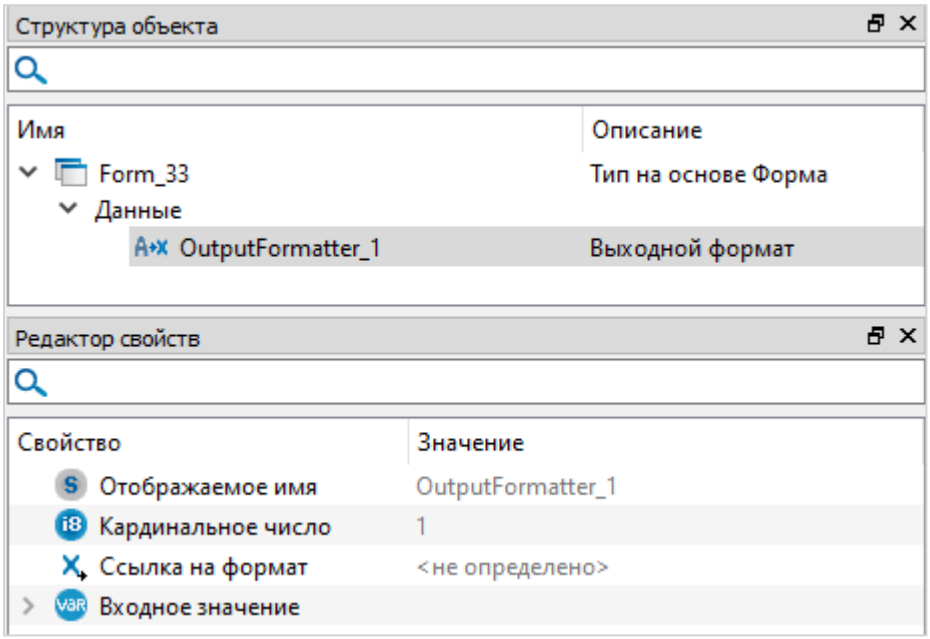


Преобразование выполняются в соответствии с форматом, указанным в свойстве Ссылка на формат (указывается один из типов **om.automation.controls**). Свойство InValue принимает входное значение строкового типа. Свойство OutValue предоставляет сконвертированное значение для чтения.

Свойство	Тип	Описание
Входное значение	string	Строка, которую нужно преобразовать к типу variant.
Выходное значение (OutValue)	variant	Значение, преобразованное к типу variant.

21.2.2. Конвертация любых типов в строки

Чтобы конвертировать значения любых типов в строковый тип string, используйте компонент **Выходной формат**. Добавьте на экранную форму компонент **Выходной формат**. Компонент невизуальный и виден только в области **Структура объекта**.



Преобразование выполняются в соответствии с форматом, указанным в свойстве Ссылка на формат (указывается один из типов **om.automation.controls**). Свойство InValue принимает входное значение любого типа. Свойство OutValue предоставляет сконвертированное значение для чтения в формате строки.

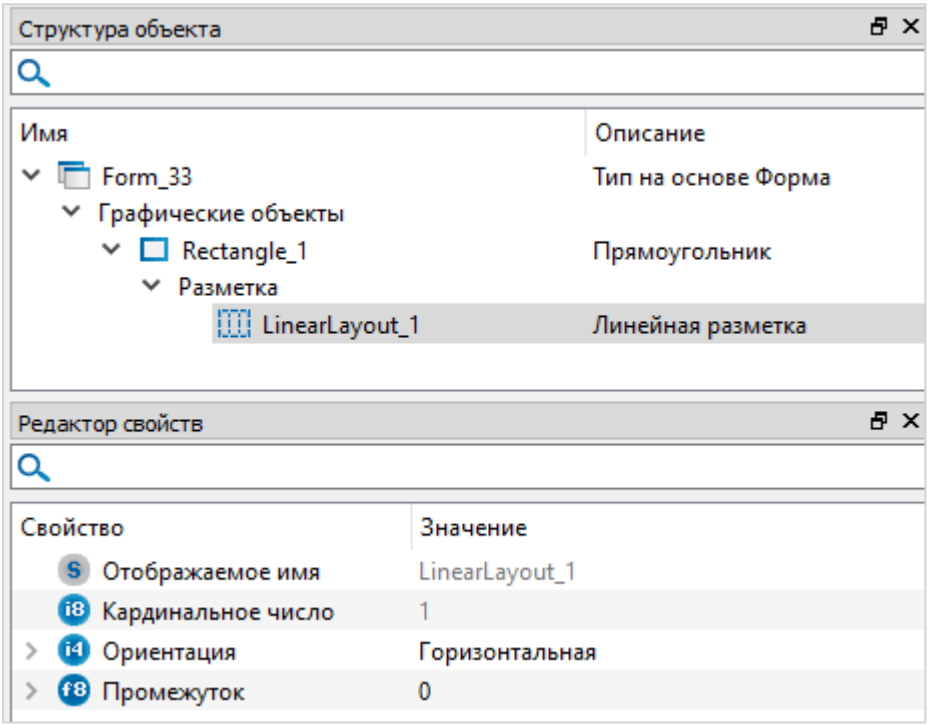
Свойство	Тип	Описание
Входное значение	variant	Значение, которое нужно преобразовать к типу string.
Выходное значение (OutValue)	string	Значение, преобразованное к типу string.

22. Разметка мнемосхемы

SePlatform.HMI предоставляет набор компонентов для разметки элементов мнемосхемы.

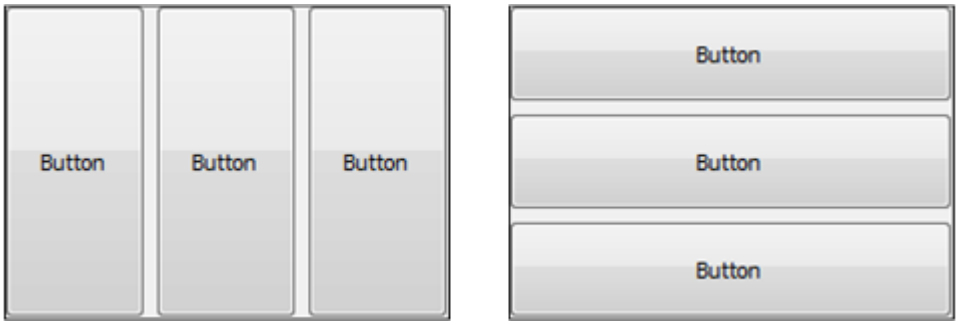
22.1. Линейная разметка

Чтобы линейно выравнять объекты внутри контейнера (экранной формы или фигуры), добавьте в контейнер компонент **Линейная разметка**. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на прямоугольник добавлен компонент **Линейная разметка**.

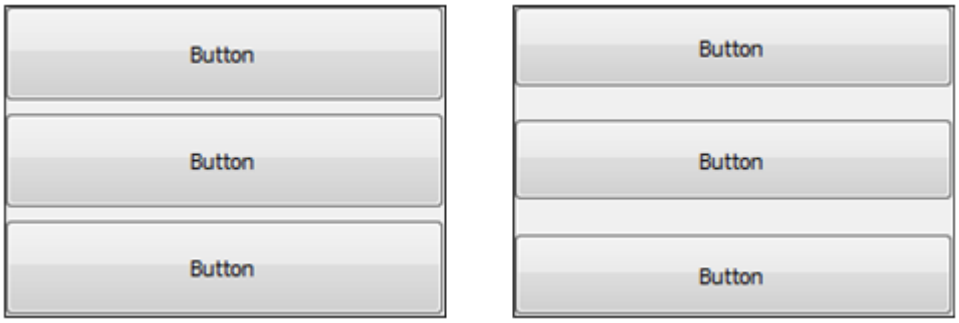


Все элементы, входящие в прямоугольник будут выравнены линейно по горизонтали или вертикали в зависимости от настройки свойств компонента:

- Ориентация (int4) - направление выравнивания объектов внутри фигуры или формы:
  - **«1»** - по горизонтали
  - **«2»** - по вертикали

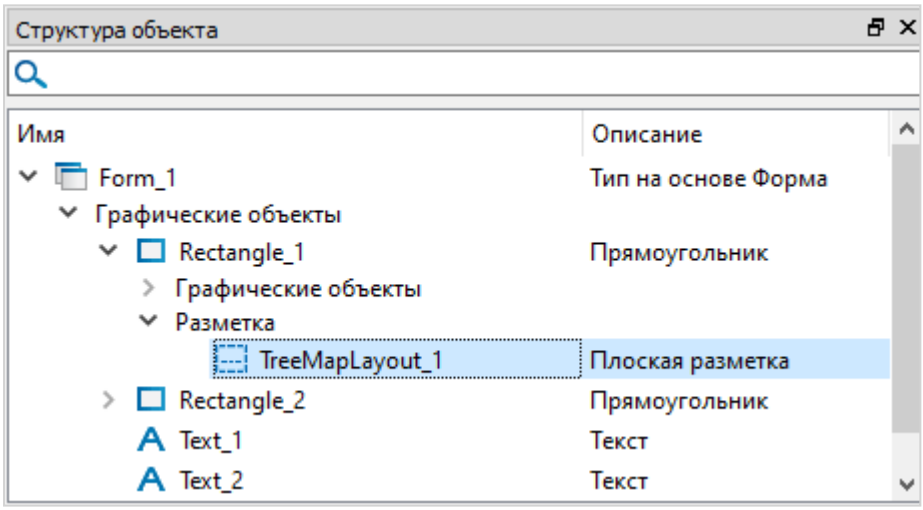


- **Промежуток** (double) - расстояние между объектами внутри фигуры или формы.



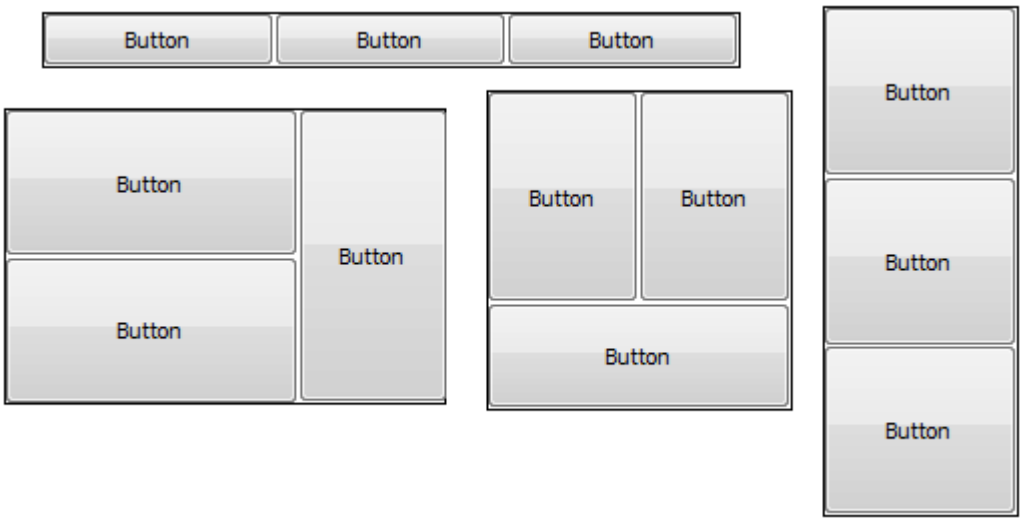
22.2. Плоская разметка

Чтобы расположить объекты внутри фигуры или экранной формы в виде вложенных объектов, используйте компонент **Плоская разметка**. Компонент необходимо использовать именно на конкретную фигуру или форму, внутри которой необходимо разместить объекты в виде вложенных объектов. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на прямоугольник добавлен компонент **Плоская разметка**.



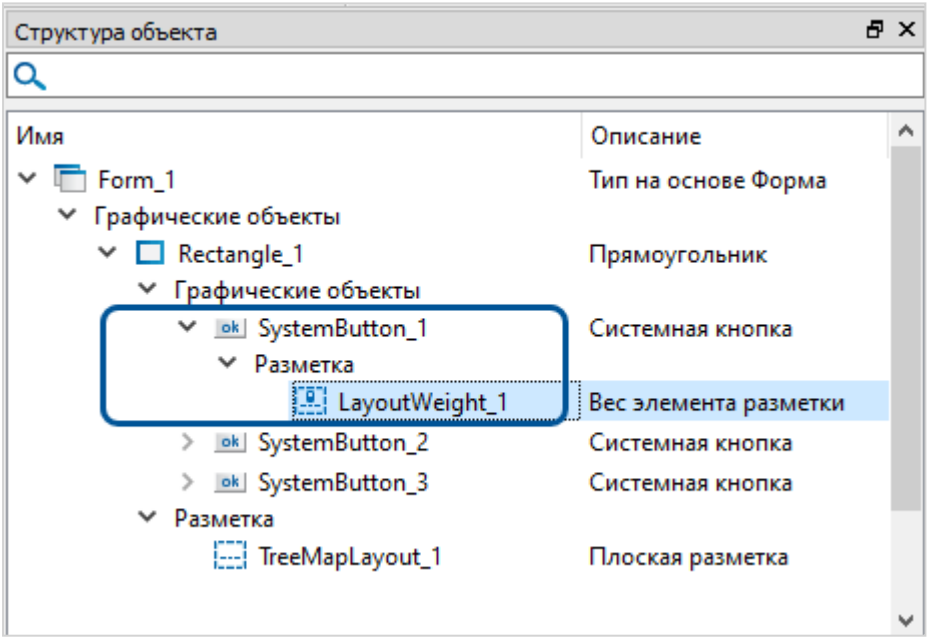
Компонент позволяет расположить объекты внутри фигуры или формы в виде вложенных объектов. Соотношения сторон между фигурой и вложенными объектами оптимизировано, т.е. размер и расположение вложенных объектов зависит от размера и расположения фигуры, в которую объекты добавлены. Компонент эффективно распределяет свободное пространство фигуры, что позволяет корректно отображать большое количество вложенных объектов.

Ниже показаны несколько вариантов плоской разметки объектов внутри фигуры.



22.3. Использование веса элемента разметки

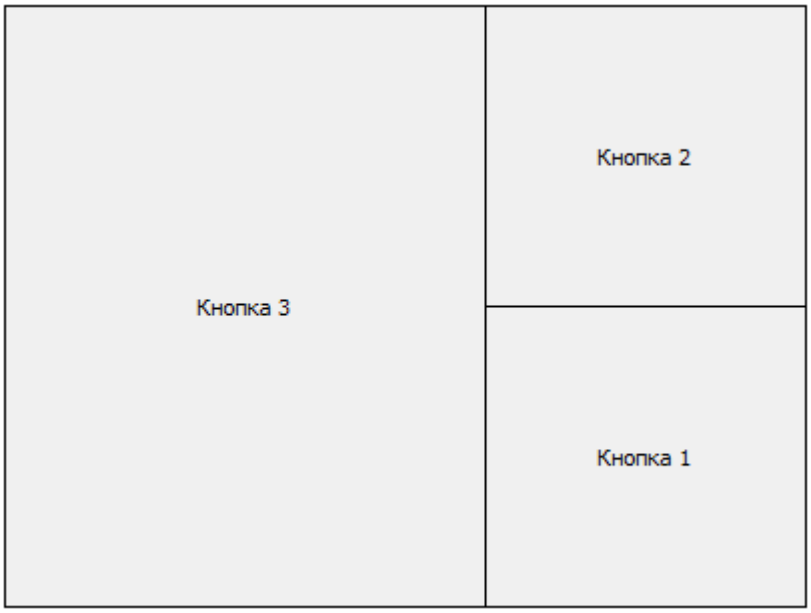
Чтобы задать объектам разметки приоритет расположения и размера по отношению к другим объектам, добавьте на фигуру компонент **Вес элемента разметки**.



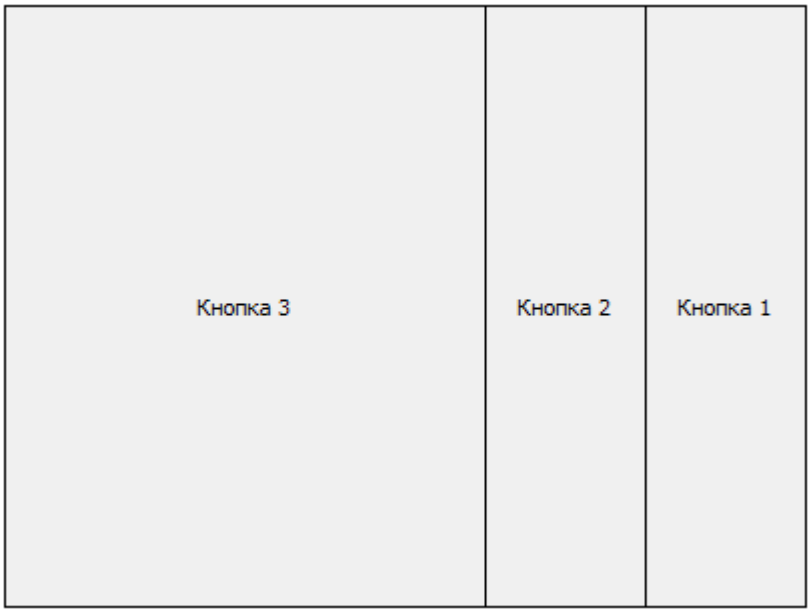
На рисунке ниже представлены примеры того, как компонент **Вес элемента разметки** влияет на распределение пространства и расположение элементов внутри контейнера для линейной и плоской разметок.

Соотношение объектов 1:3. Свойство Значимость для первой кнопки «1», для второй -«1», для третьей - «3».

Плоская разметка с использованием веса элементов



Линейная разметка с использованием веса элементов

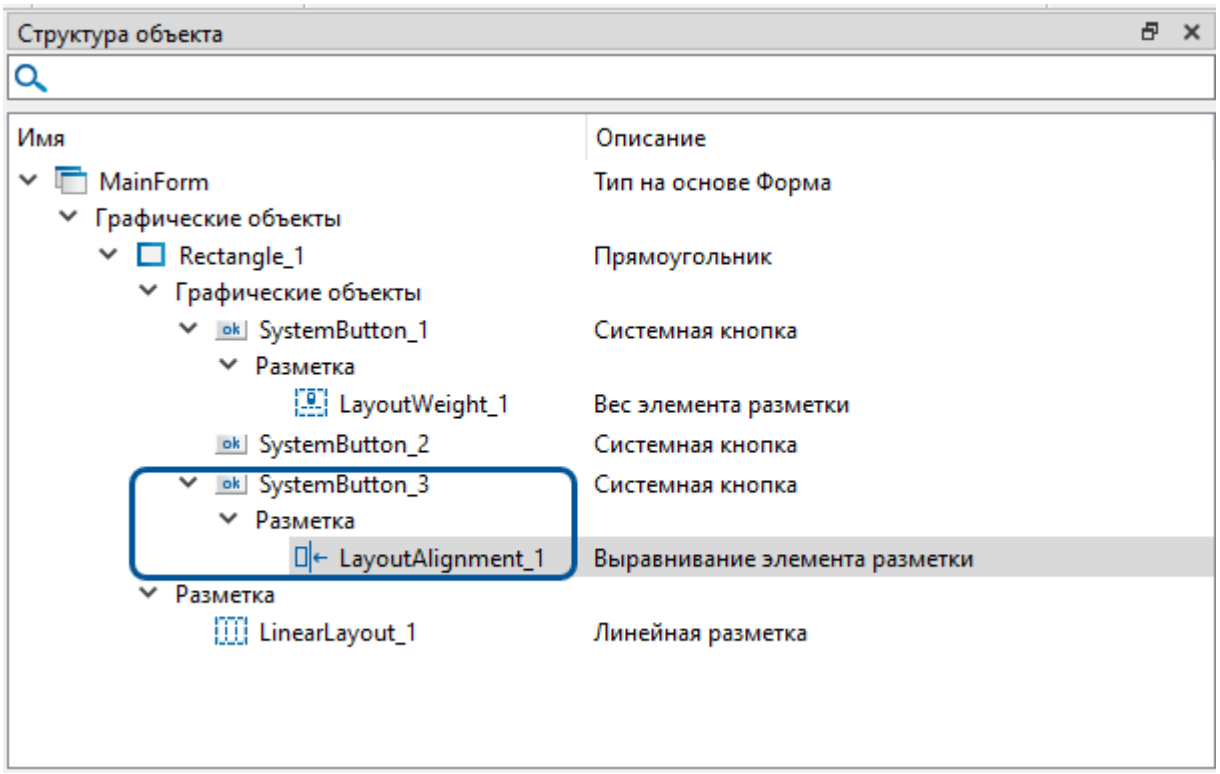


- Плоская размета: В этом случае вес элемента прямо пропорционален его размеру и влияет на его расположение внутри контейнера.
- Линейная разметка: В этом случае вес элемента определяет размер элемента, не изменяя порядок расположения элементов.

Чтобы настроить вес элемента в разметке, установите свойство **Значимость**.

22.4. Выравнивание элемента разметки

Чтобы настроить выравнивание элемента линейной разметки отдельно для горизонтальной и вертикальной осей, добавьте на фигуру компонент **Выравнивание элемента разметки**.

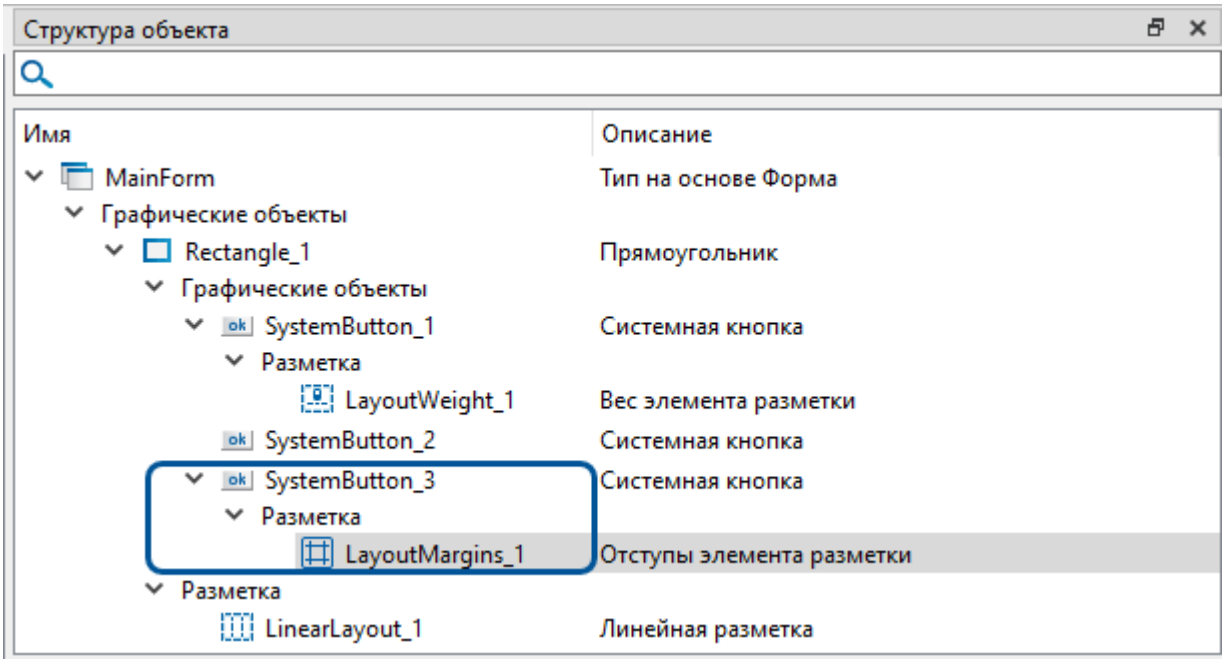


Для настройки выравнивания элемента в разметке:

- Горизонтальное выравнивание: Используйте свойство **Выравнивание по горизонтали**, чтобы выровнять элемент в линейной разметке слева, по центру или справа.
- Вертикальное выравнивание: Используйте свойство **Выравнивание по вертикали**, чтобы выровнять элемент в линейной разметке сверху, по центру или снизу.

22.5. Отступы элемента разметки

Чтобы установить отступы вокруг элемента линейной разметки, добавьте на фигуру компонент **Отступы элемента разметки**.

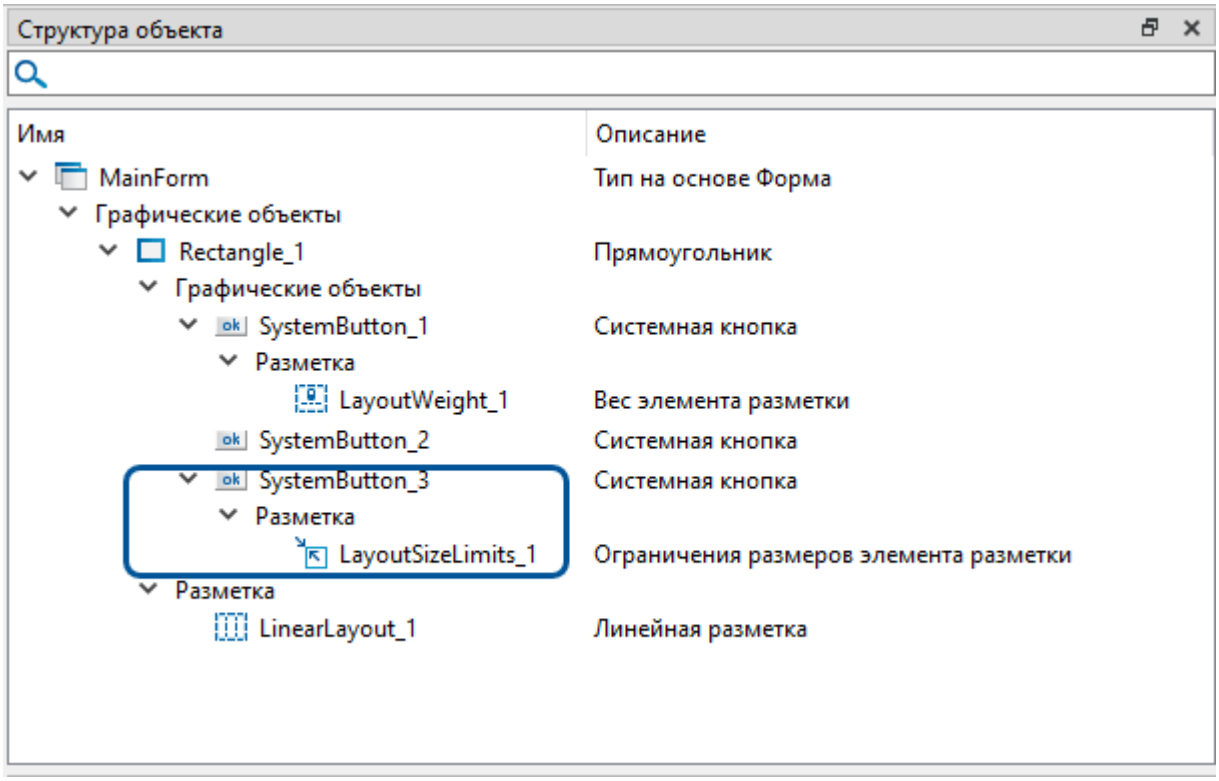


Чтобы настроить отступы для элемента разметки используйте свойства:

- **Сверху** - для установки верхнего отступа.
- **Снизу** - для установки нижнего отступа.
- **Слева** - для установки левого отступа.
- **Справа** - для установки правого отступа.

22.6. Ограничение размеров элемента разметки

Чтобы задать минимальную и максимальную ширину и высоту элемента линейной разметки, добавьте на фигуру компонент **Ограничение размеров элемента разметки**.



Для задания ограничений размеров элемента в разметке:

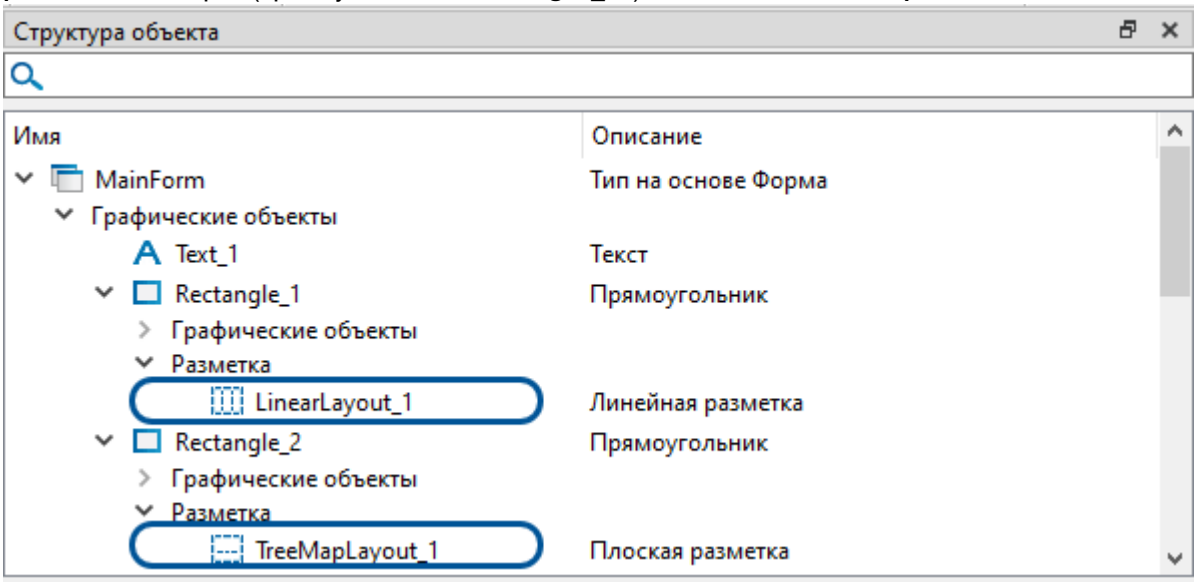
- Используйте свойства **Минимальная ширина** и **Минимальная высота**, чтобы установить минимальные размеры элемента.
- Используйте свойства **Максимальная ширина** и **Максимальная высота**, чтобы установить предельные максимальные размеры.

22.7. Демо-приложение Применение разметки

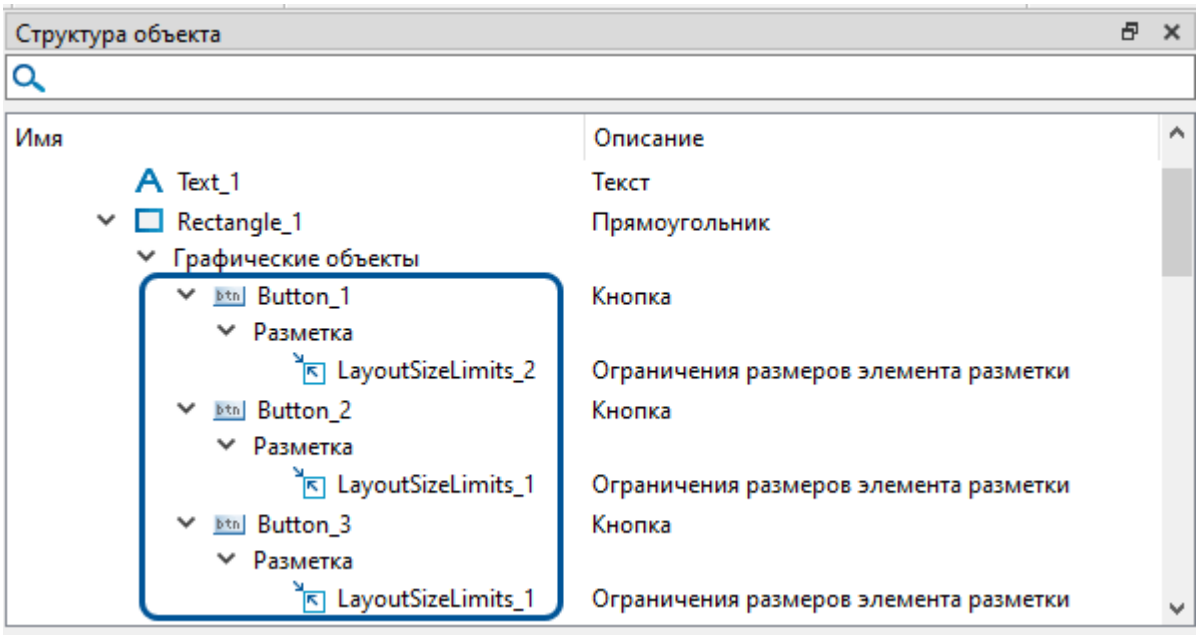
Демо-приложение **«Применение разметки»** иллюстрирует использование компонентов разметки мнемосхемы, показывая изменение ориентации и промежутков между объектами в линейной разметке, установку значимости в плоской разметке, а также возможности контроля за размерами, выравниванием и настройкой отступов элементов.

Ниже показан пользовательский интерфейс окна приложения.

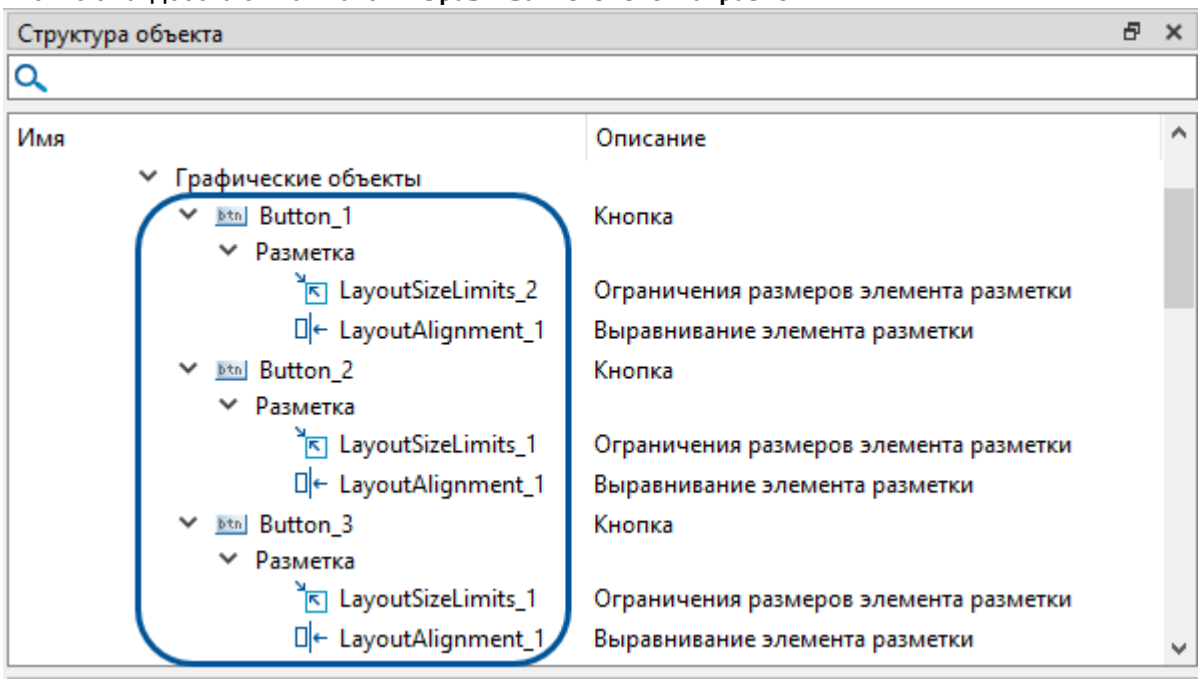
Для разметки кнопок на первый контейнер (прямоугольник «**Rectangle_1**») добавлен компонент **Линейная разметка**, на второй (прямоугольник «**Rectangle_2**»)- компонент **Плоская разметка**.



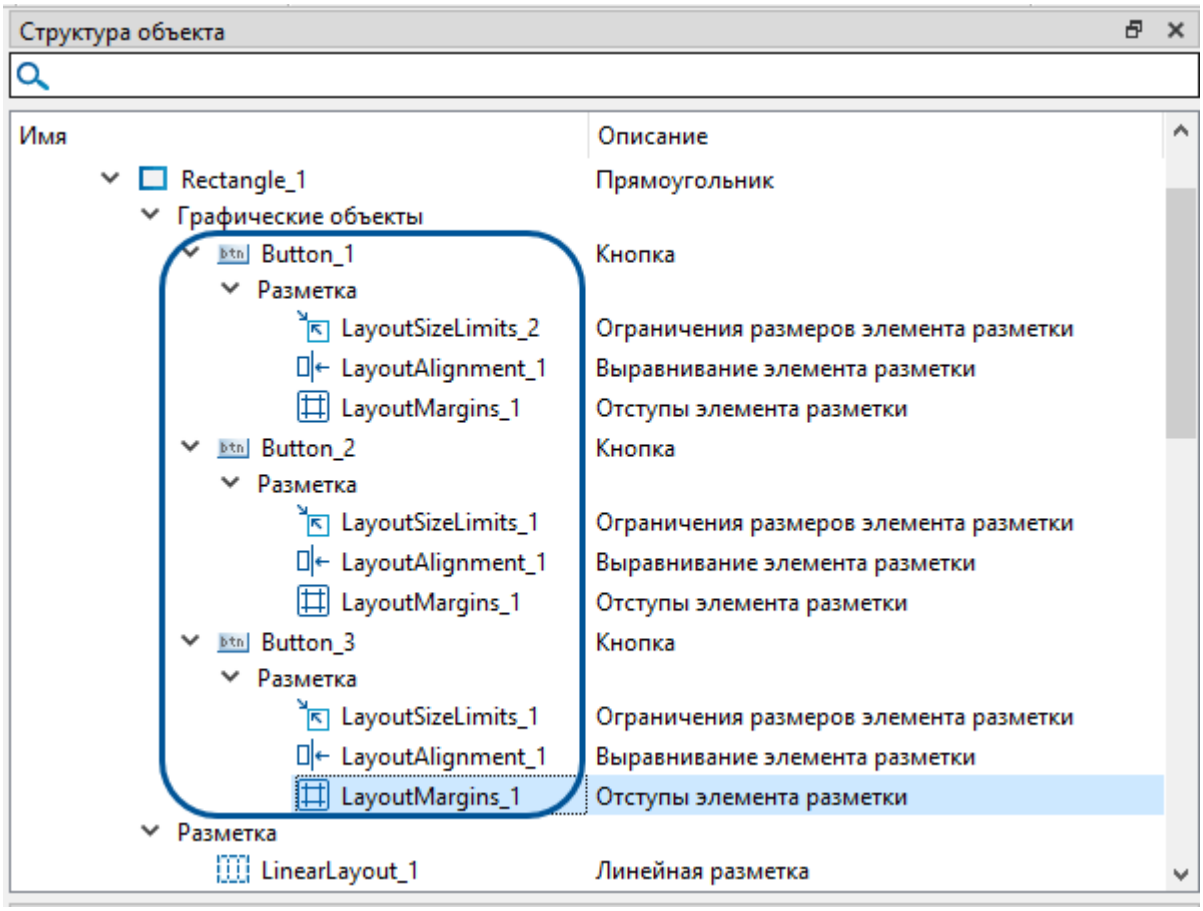
Чтобы задать минимальную ширину и максимальную высоту элементам первого контейнера, на каждую кнопку добавлен компонент **Ограничения размеров элемента разметки**.



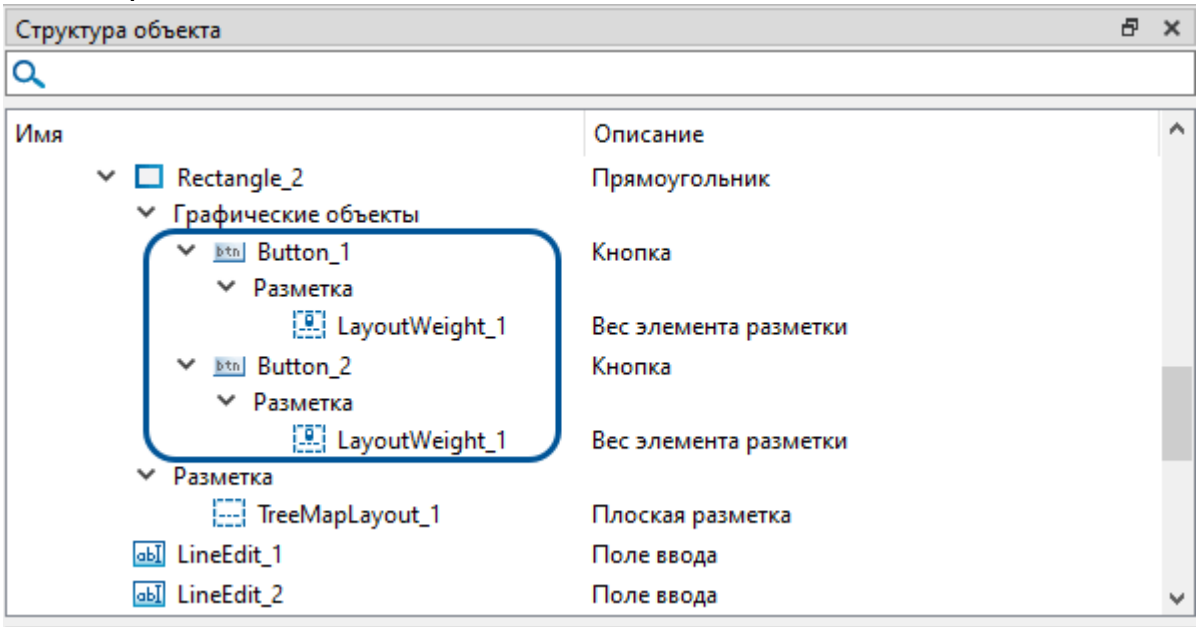
Для горизонтального выравнивания элементов первого контейнера по левому или правому краю, каждой кнопке был добавлен компонент **Выравнивание элемента разметки**.



Для установки левого и правого отступов элементов в первом контейнере, к каждой кнопке был добавлен компонент **Отступы элемента разметки**.



Чтобы установить вес разметки кнопкам второго контейнера, на каждую кнопку добавлен компонент **Вес элемента разметки**.



Чтобы элементы первого контейнера выстроились в горизонтальной ориентации, для кнопки **Горизонтальная разметка** создайте обработчик событий. Код обработчика на языке SePlatform.Om:

```
Rectangle_1.LinearLayout_1.Orientation = 1;
```

Чтобы элементы первого контейнера выстроились в вертикальной ориентацию, для кнопки **Вертикальная разметка** создайте обработчик событий. Код обработчика на языке SePlatform.Om:

```
Rectangle_1.LinearLayout_1.Orientation = 2;
```

Чтобы промежуток между элементами в первом прямоугольнике принял значение 50px и 20px, для кнопок **Промежуток 50px** и **Промежуток 20px** создайте обработчики событий. Код обработчиков на языке SePlatform.Om:

```
Rectangle_1.LinearLayout_1.Gap = 50;
Rectangle_1.LinearLayout_1.Gap = 20;
```

Чтобы выравнять элементы в первом прямоугольнике по левому или правому краю, для кнопок **Выравнять по левому краю** и **Выравнять по правому краю** создайте обработчики событий. Код обработчиков на языке SePlatform.Om:

```
// Выравнивание по левому краю
Rectangle_1.Button_1.LayoutAlignment_1.HorizontalAlignment = 0;
Rectangle_1.Button_2.LayoutAlignment_1.HorizontalAlignment = 0;
Rectangle_1.Button_3.LayoutAlignment_1.HorizontalAlignment = 0;
// Выравнивание по правому краю
Rectangle_1.Button_1.LayoutAlignment_1.HorizontalAlignment = 2;
Rectangle_1.Button_2.LayoutAlignment_1.HorizontalAlignment = 2;
Rectangle_1.Button_3.LayoutAlignment_1.HorizontalAlignment = 2;
```

Чтобы элементам первого контейнера установить ограничение размеров, для кнопки **Применить ограничения** создайте обработчик событий. Обработчик конвертирует и передает значение из текстового поля в значение свойства **Минимальная ширина** и **Максимальная высота**. Код обработчика на языке SePlatform.Om:

```
// Ограничение минимальной ширины
Rectangle_1.Button_1.LayoutSizeLimits_2.MinWidth = String.ToDouble(LineEdit_5.Text,1);
Rectangle_1.Button_2.LayoutSizeLimits_1.MinWidth = String.ToDouble(LineEdit_5.Text,1);
Rectangle_1.Button_3.LayoutSizeLimits_1.MinWidth = String.ToDouble(LineEdit_5.Text,1);
// Ограничение максимальной высоты
Rectangle_1.Button_1.LayoutSizeLimits_2.MaxHeight = String.ToDouble(LineEdit_6.Text,1);
Rectangle_1.Button_2.LayoutSizeLimits_1.MaxHeight = String.ToDouble(LineEdit_6.Text,1);
Rectangle_1.Button_3.LayoutSizeLimits_1.MaxHeight = String.ToDouble(LineEdit_6.Text,1);
```

Чтобы элементам первого контейнера задать отступы вокруг элементов, для кнопки **Применить отступы** создайте обработчик событий. Обработчик конвертирует и передает значение из текстового поля в значение свойства **Слева** и **Справа**. Код обработчика на языке SePlatform.Om:

```
// Отступы слева
Rectangle_1.Button_1.LayoutMargins_1.Left = String.ToDouble(LineEdit_7.Text,1);
Rectangle_1.Button_2.LayoutMargins_1.Left = String.ToDouble(LineEdit_7.Text,1);
Rectangle_1.Button_3.LayoutMargins_1.Left = String.ToDouble(LineEdit_7.Text,1);
// Отступы справа
Rectangle_1.Button_1.LayoutMargins_1.Right = String.ToDouble(LineEdit_8.Text,1);
Rectangle_1.Button_2.LayoutMargins_1.Right = String.ToDouble(LineEdit_8.Text,1);
Rectangle_1.Button_3.LayoutMargins_1.Right = String.ToDouble(LineEdit_8.Text,1);
```

Чтобы элементам второго контейнера установить значимость разметки, для кнопки **Применить значимость элементам** создайте обработчик событий. Обработчик конвертирует и передает значение из текстового поля в значение свойства **Значимость**. Код обработчика на языке SePlatform.Om:

```
Rectangle_2.Button_1.LayoutWeight_1 = String.ToDouble(LineEdit_1.Text,1);
Rectangle_2.Button_2.LayoutWeight_1 = String.ToDouble(LineEdit_2.Text,1);
```

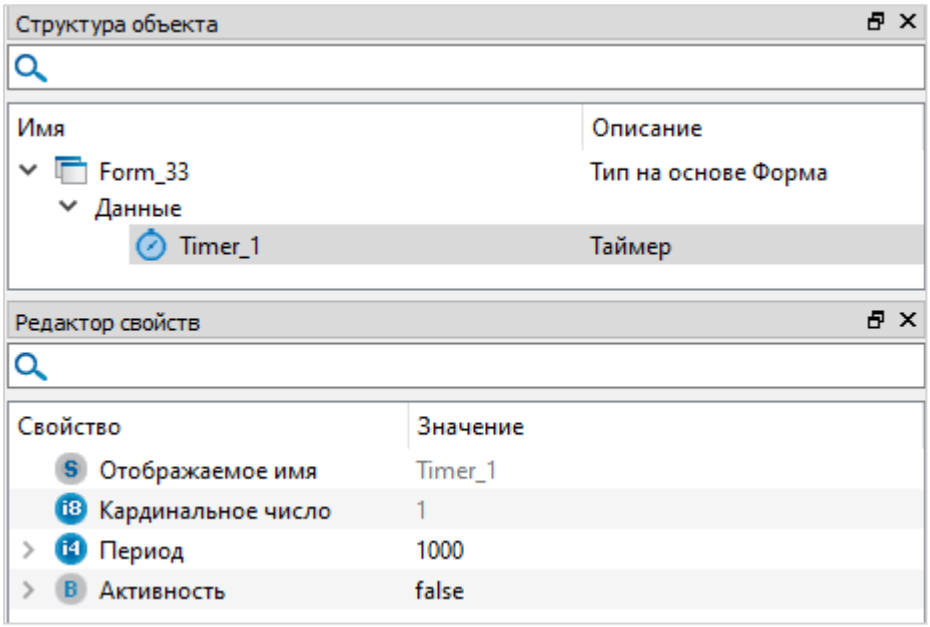
23. Динамика и проигрывание звуков

SePlatform.HMI предоставляет набор компонентов для применения динамики и проигрывания звуков на мнемосхеме. Под динамикой подразумевается:

- мигание графических объектов, например задвижек;
- циклическое выполнение каких-либо процедур, например обновление локального времени.

23.1. Выполнение процедур по таймеру

Чтобы циклично выполнять процедуру с заданной периодичностью, добавьте на фигуру или экранную форму компонент **Таймер**. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на экранную форму добавлен компонент **Таймер**.



Компонент позволяет циклически выполнять процедуру (определяется в обработчике события OnTimer), которая повторяются с заданной периодичностью (свойство Период).

Свойства

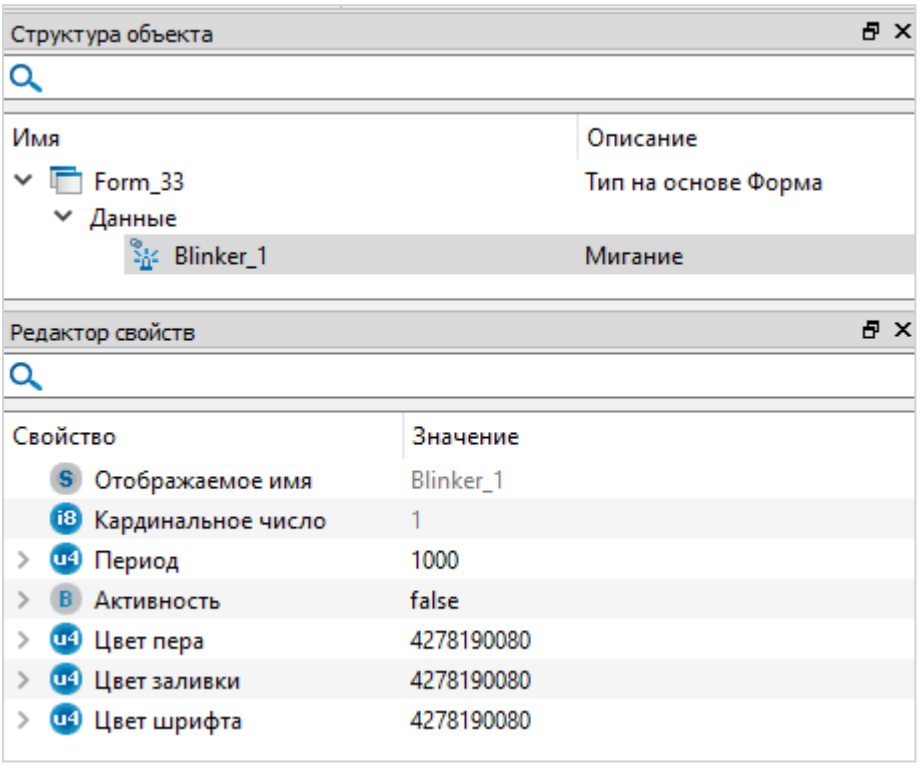
Свойство	Тип	Описание
Период	int4	Период исполнения события по таймеру.
Активность	bool	Активность таймера. Если таймер неактивен, то событие по таймеру не будет исполняться.

События

Событие	Описание
OnTimer	Событие, выполняемое по таймеру с заданным Периодом. Параметры: - time (тип timestamp) - Текущее время UTC.

23.2. Мигание графических объектов

Чтобы графические объекты получили возможность мигания, добавьте на фигуру или экранную форму компонент **Мигание**. Компонент невизуальный и виден только в области **Структура объекта**. На рисунке ниже показано, как на экранную форму был добавлен компонент **Мигание**.



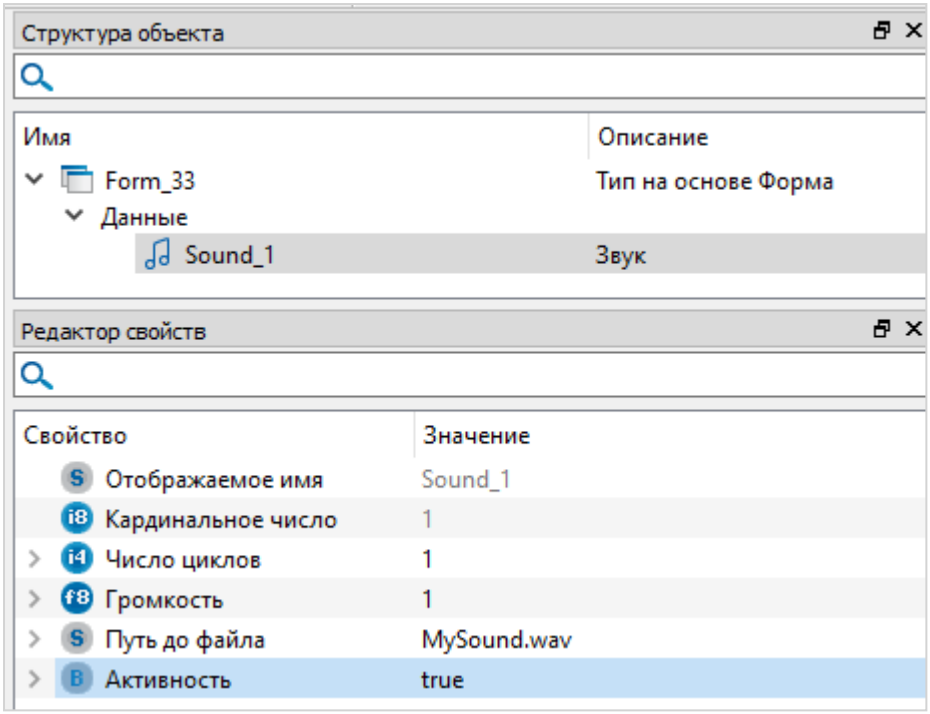
Компонент позволяет графическим компонентам циклически с заданной периодичностью (свойство Период) менять цвет заливки (свойство Цвет заливки) или внешней границы (свойство Цвет пера). Цвет попеременно изменяется от стандартного цвета графического компонента к цвету, который настроен в свойствах компонента **Мигание**.

Свойства

Свойство	Тип	Описание
Период	uint4	Период мигания графического объекта.
Активность	bool	Активность мигания. Если мигание неактивно, то мигание не будет исполняться.
Цвет пера	uint4	Цвет внешней границы мигающего объекта. При мигании объекта цвет внешней границы меняется с цвета, указанного в свойствах объекта, на цвет, заданный в свойствах объекта **Мигание**.
Цвет заливки	uint4	Цвет внутренней заливки мигающего объекта. При мигании объекта цвет заливки меняется с цвета, указанного в свойствах объекта, на цвет, заданный в свойствах объекта **Мигание**.

23.3. Проигрывание звуков

Чтобы добавить звуки на мнемосхему, добавьте компонент **Звук**. Компонент невизуальный и виден только в области **Структура объекта**.



После добавления компонента **Звук** на экранную форму, укажите в свойстве Путь до файла название файла в формате `wav`, который следует проигрывать. Звуковой файл должен находиться в папке проекта `resources`. Чтобы звук начал проигрываться в режиме исполнения, переключите свойство Активность в состояние **«true»**.
Для проигрывания очереди звуков используйте обработчики событий LoopPlayed (завершение одного цикла проигрывания звука) или AllLoopsPlayed (завершение всех циклов проигрывания звука).

23.4. Демо-приложение Очередь звуков

Ниже приведен пример настройки очереди звуков для проигрывания:

1. В папке `resources` проекта находятся три звуковых файла `Kick 1.wav`, `Kick 2.wav`, `Kick 3.wav`.
2. На форму добавлен звук **«Sound_1»**.
3. Для запуска и остановки проигрывания очереди звуков на форму добавлены кнопки **«Начать проигрывание»** и **«Остановить проигрывание»**.
4. Для учета числа срабатываний события LoopPlayed звука **«Sound_1»** в проект добавлено поле **«int4_1»**.
5. В обработчик события ButtonPressed кнопки **«Начать проигрывание»** записан код:

```
int4_1=0; //обнулить число срабатываний события LoopPlayed объекта Sound_1
Sound_1.Path="Kick 1.wav"; //указать начальный звуковой файл для проигрывания
Sound_1.Active=true; //начать проигрывание очереди звуков
```

6. В обработчик события ButtonPressed кнопки **«Остановить проигрывание»** записан код:

```
int4_1=0; //обнулить число срабатываний события LoopPlayed объекта Sound_1
Sound_1.Active=false; //остановить проигрывание очереди звуков
```

7. В обработчик события LoopPlayed звука **«Sound_1»** записан код:

```
int4_1=int4_1+1; //нарастить число срабатываний события LoopPlayed объекта Sound_1
Text_1.Text = Str.ToString(int4_1); //вывести число срабатываний события LoopPlayed
объекта Sound_1
if(int4_1==1) { //проигрывать очередной звук при каждом увеличении числа срабатываний
события LoopPlayed объекта Sound_1
  Sound_1.Path="Kick 2.wav";
  Sound_1.Active=true; }
if(int4_1==2) {
  Sound_1.Path="Kick 3.wav";
  Sound_1.Active=true; }
if(int4_1==3) {
  int4_1=0;
  Sound_1.Active=false; } //завершить проигрывание очереди звуков
```

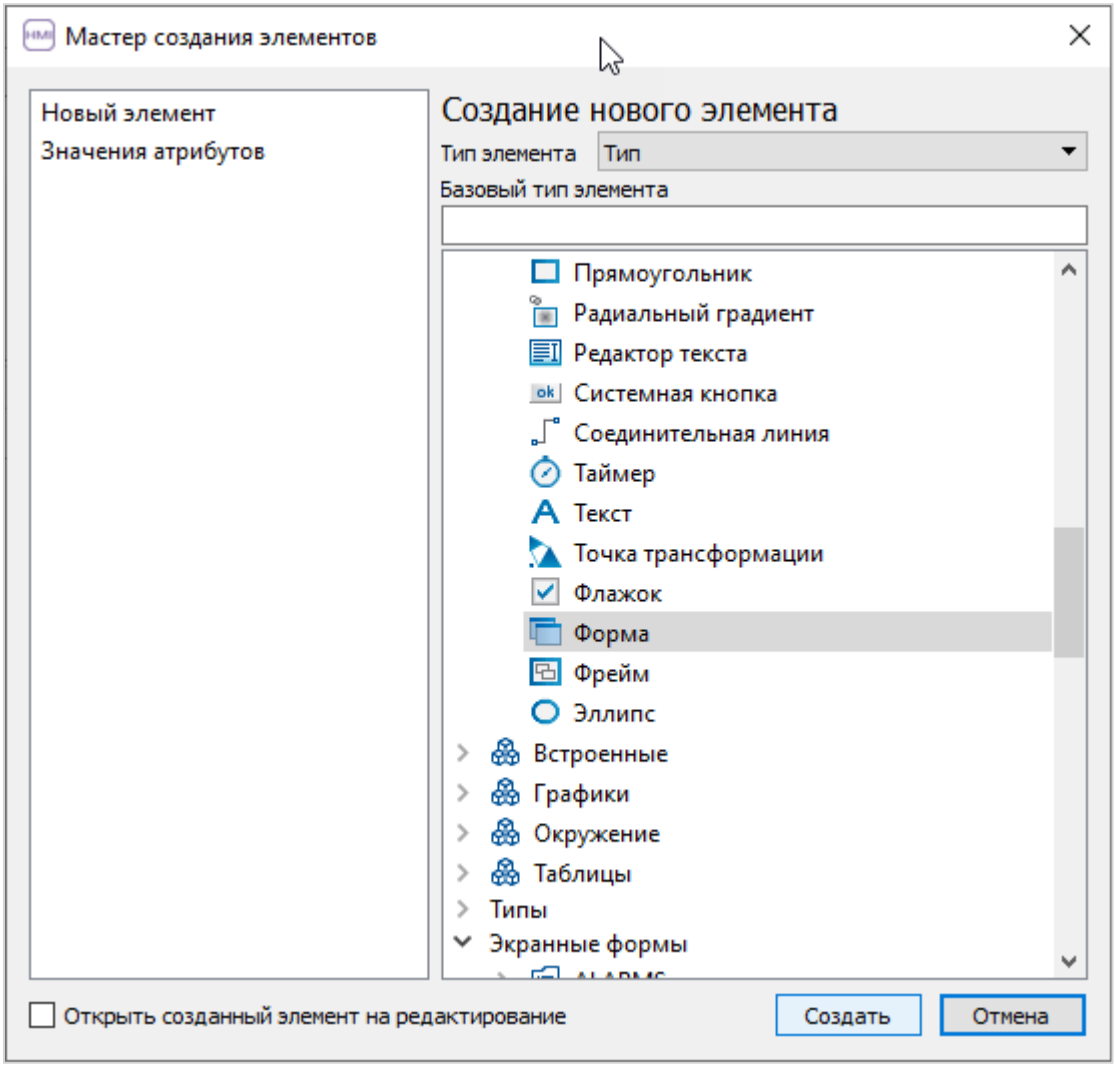
В результате, при нажатии кнопки **Начать проигрывание** будут подряд проиграны три звуковых файла (`Kick 1.wav`, `Kick 2.wav`, `Kick 3.wav`). Проигрывание завершится после окончания третьего звука. В любой момент проигрывания звуков можно нажать кнопку **Остановить проигрывание**.

24. Работа с экранными формами

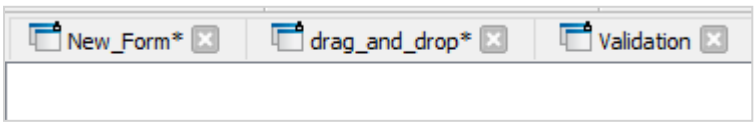
SePlatform.HMI позволяет разрабатывать многооконные проекты. Для этого добавляйте в проект новые экранные формы.

24.1. Создание экранных форм

Чтобы создать новую экранную форму, выберите команду контекстного меню **Создать** и воспользуйтесь мастером создания элементов.

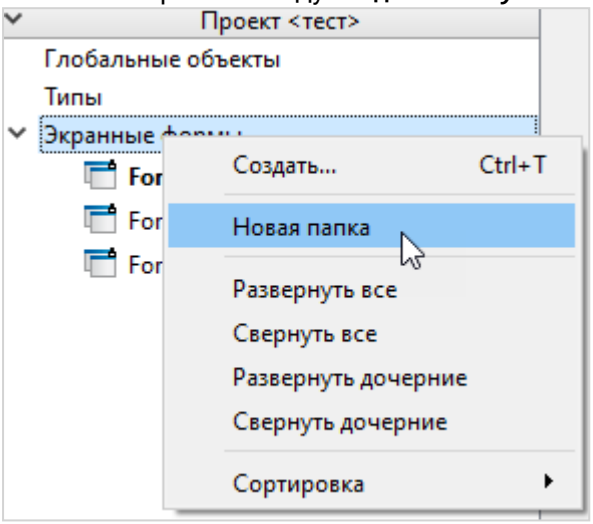


Создастся новая экранная форма со стандартным названием. Для каждой экранной формы создается новая вкладка рабочей области. Чтобы быстро переключаться между открытыми вкладками рабочей области, используйте сочетание клавиш «**Ctrl**»+«**Tab**».



Чтобы открыть экранную форму для редактирования, дважды щелкните по ней в библиотеке компонентов.

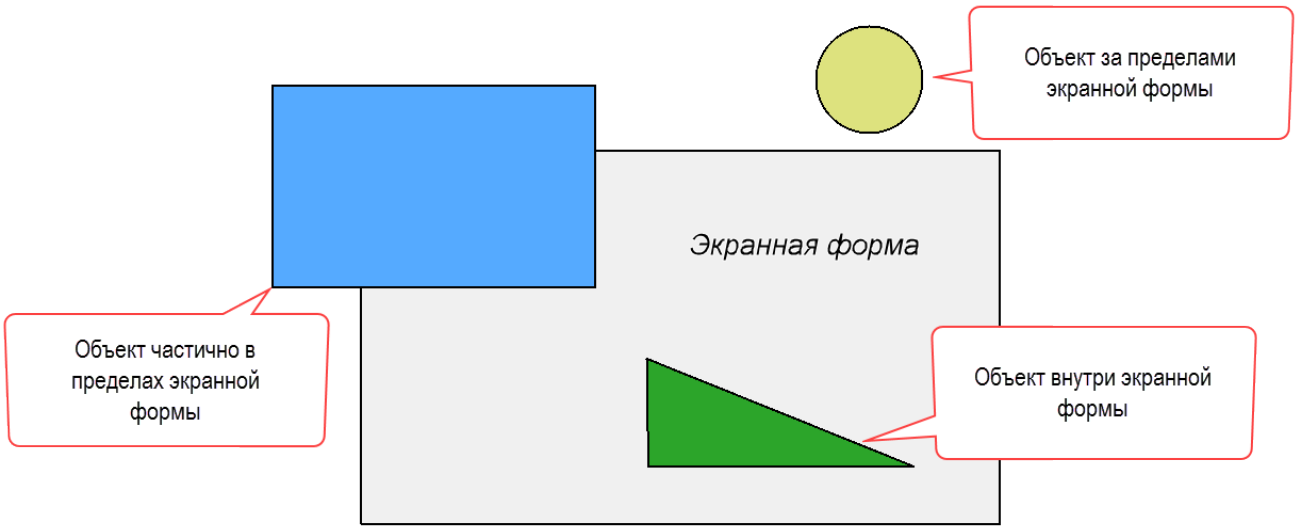
При первичном редактировании формы появится чистая рабочая область. Возможности работы в рабочей области экранной формы были описаны ранее (стр. 1).
Чтобы создать экранную форму на основе уже имеющейся экранной формы, выберите команду контекстного меню **Создать экранную форму** → **На основе** → **Проект <Имя проекта>** → **<Имя формы>**.
Чтобы скопировать экранную форму вместе с содержимым, в **Библиотеке компонентов** выберите экранную форму и команду **Копировать тип** в контекстном меню формы. Копия добавится рядом с копируемой формой.
Чтобы группировать экранные формы, используйте папки. Для создания папки перейдите в контекстное меню и выберите команду **Создать папку**.



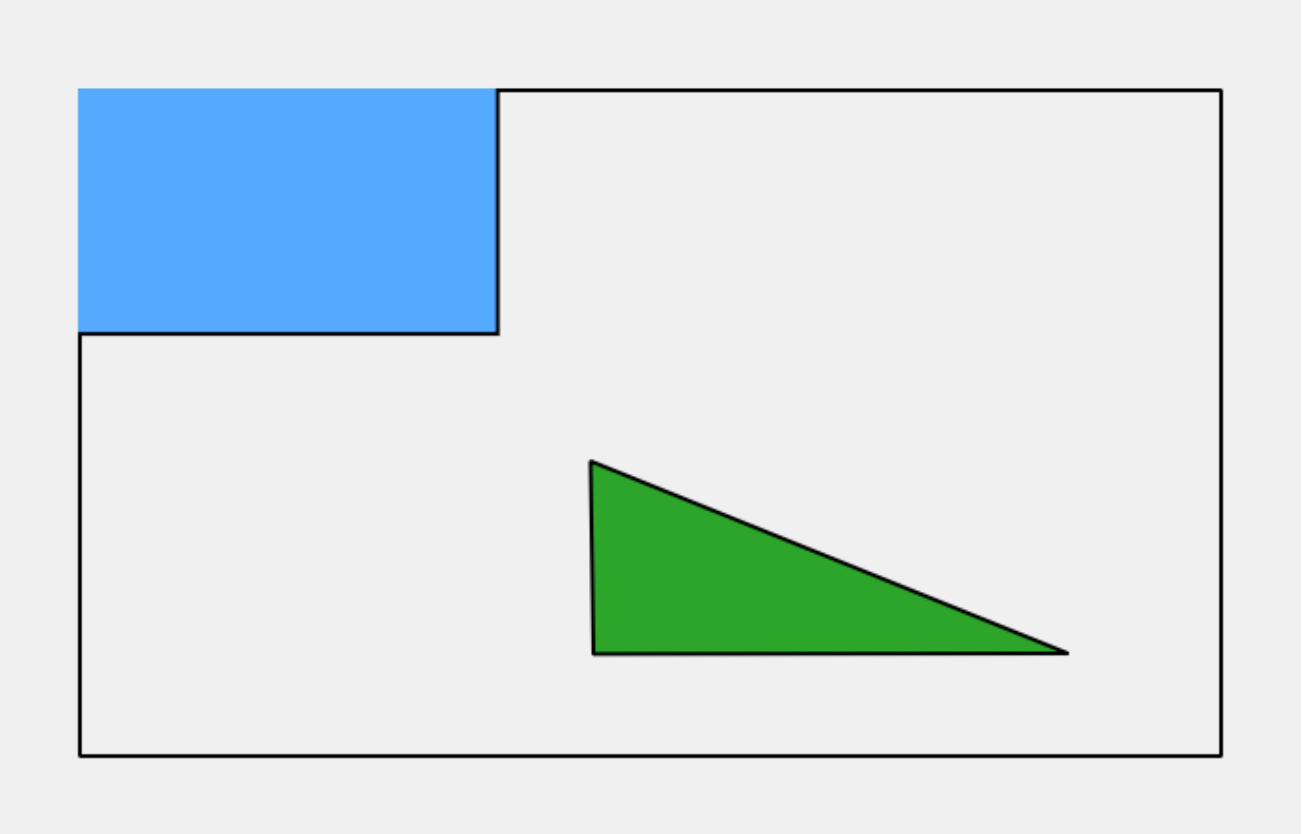
Созданные экранные формы сохраняются в папке проекта `objects` с расширением файлов `*.omobj`.
Созданные экранные формы можно экспортировать (стр. 59) из проекта или импортировать (стр. 59) в проект.
Чтобы настроить переходы между экранными формами, настройте обработчики событий:

- открыть в текущем окне
- открыть в новом окне
- открыть в диалоговом окне
- открыть во фрейме

Проектируя визуальную часть проекта, помните, что после запуска в рантайме (стр. 16) в окне экранной формы отобразятся только те объекты, которые были расположены в пределах экранной формы.



После запуска в рантайме вы получите следующий результат.



24.2. Указание главной формы

Чтобы указать главную форму в проекте, вызовите контекстное меню нужной экранной формы и выберите команду **Установить форму главной**.

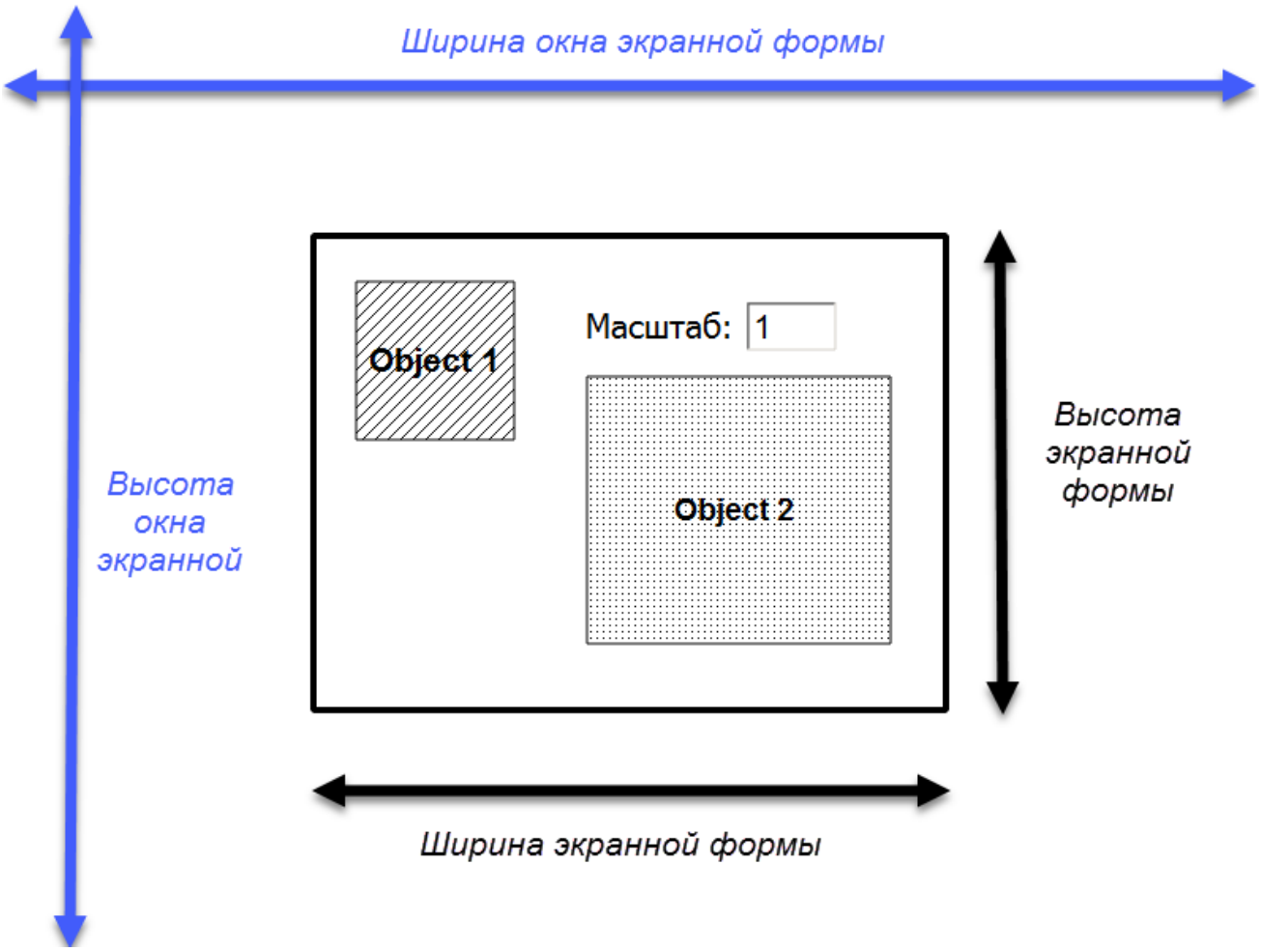
Главная форма запускается в рантайм при:

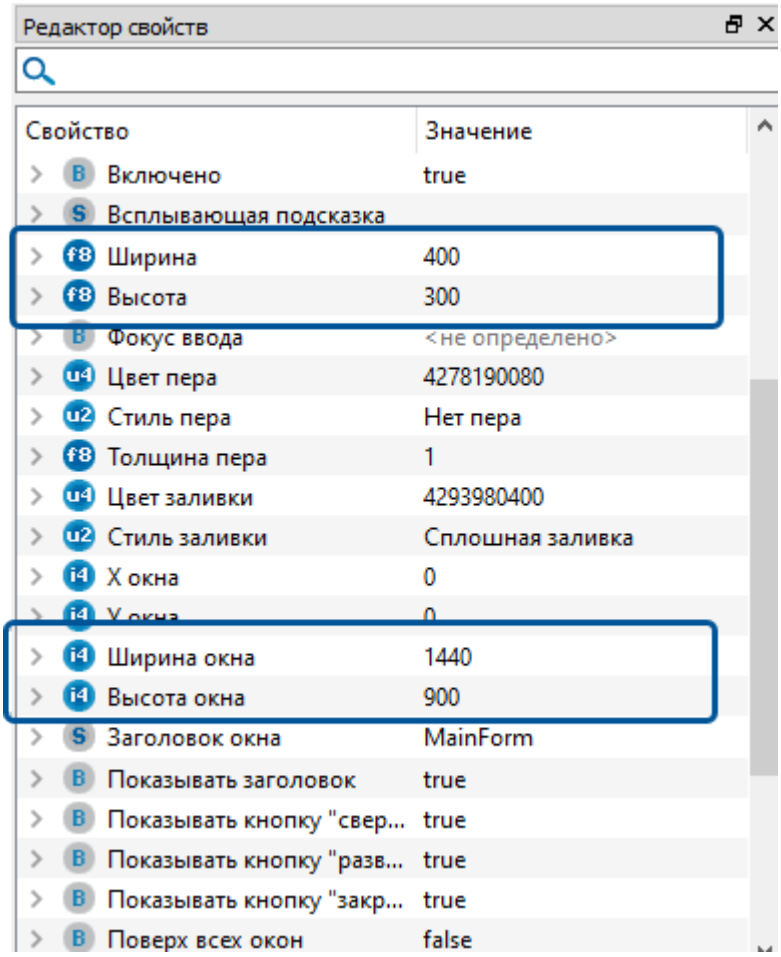
- запуске проекта из проводника Windows (стр. 19);
- выборе команды **Показать главную форму в рантайме** или нажатии клавиши **«F9»** в Дизайнере SePlatform.HMI (стр. 16).

Если ни одна форма не назначена главной, новая форма при добавлении автоматически устанавливается главной.

24.3. Размер и формат экранных форм

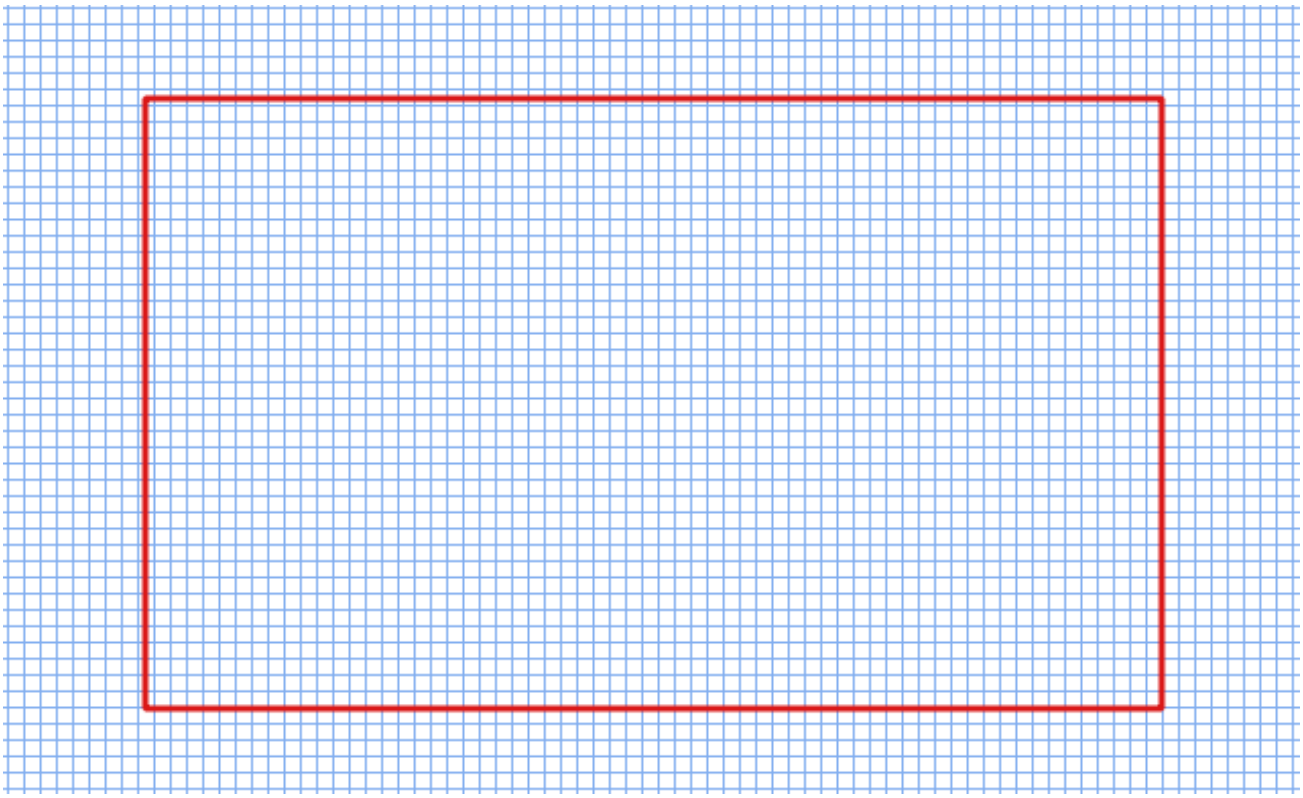
Любая экранная форма при запуске в рантайме будет расположена внутри окна-контейнера. На схеме ниже показана экранная формы внутри окна, и свойства, которые определяют размер формы и окна. Все свойства расположены в едином редакторе свойств.


Ширина окна экранной формы
Высота окна экранной
Object 1
Масштаб: 1
Object 2
Высота экранной формы
Ширина экранной формы

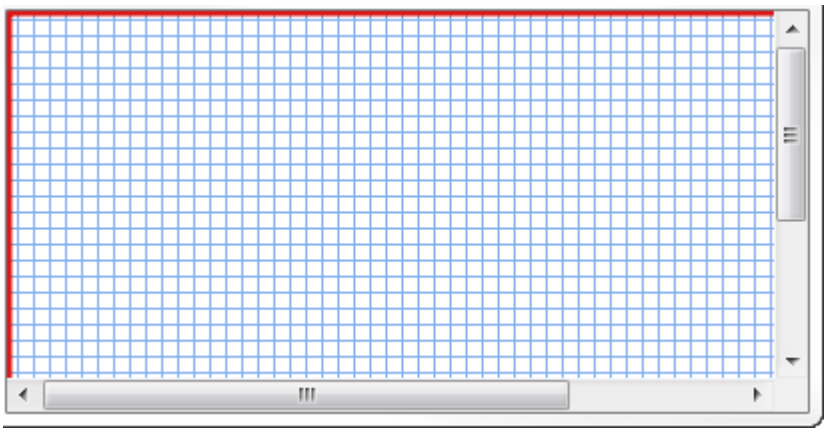


Размеры окна могут совпадать или не совпадать с размерами экранной формы:

- если размер окна больше экранной формы, то будут видны пустые области.



› если размер окна меньше экранной формы, то появятся полосы прокрутки.



Варианты установки размеров окна:

› Чтобы указать точный размер окна вручную (в пикселях), установите свойство экранной формы **Размеры окна** в состояние **«Вручную»** и укажите значения свойств **Ширина окна** и **Высота окна**.

› Чтобы размер окна при открытии автоматически подстроился под размер экранной формы (и в дальнейшем не менялся), установите свойство **Размеры окна** в положение **«Автоподбор при открытии»**.

› Чтобы размер окна динамически подстраивался под размер экранной формы на протяжении всего времени работы с окном, установите свойство **Размеры окна** в положение **«Автоподбор»**.

> ПРИМЕЧАНИЕ
>
> Режим **«Автоподбор»** может быть полезен, если размеры экранной формы динамически меняются в процессе исполнения мнемосхемы.

› Чтобы размер окна автоматически подстроился под размеры экранной формы в режиме исполнения мнемосхемы, воспользуйтесь функцией экранной формы **DoAutoSize**.

› Чтобы заблокировать возможность менять размеры окна, выставьте свойству **Стиль рамки окна** значение **«Фиксированный размер»**.

› Чтобы заблокировать возможность перемещать окно и менять размеры окна, выставите свойству **Стиль рамки окна** значение **«Без рамки»**.

Варианты установки размеров и масштаба экранной формы:

› Чтобы указать вручную точный размер экранной формы (в пикселях), установите свойство **Режим масштабирования** в состояние **«Не масштабировать»** и укажите значения свойств **Ширина** и **Высота**.

› Чтобы указать вручную коэффициент масштабирования экранной формы, установите свойство **Масштаб**. Значения меньше 1 уменьшают масштаб, значения больше 1 - увеличивают масштаб.

› Чтобы автоматически уменьшать масштаб экранной формы до размеров окна, установите свойство **Режим масштабирования** в положение **«Только уменьшение»**.

> ОБРАТИТЕ ВНИМАНИЕ
>
> Использование опций авто-масштабирования экранной формы максимально впишет экранную форму в окно, но может привести к плохой прорисовке графических элементов или, напротив, к излишней величине графических элементов.

› Чтобы автоматически подгонять масштаб экранной формы до размеров окна, установите свойство **Режим масштабирования** в положение **«Всегда масштабировать»**.

Варианты установки состояния окна при открытии:

› Чтобы окно открывалось в одном из системных состояний (свернуто , развёрнуто, на весь экран), выберите нужное состояние для свойства Состояние окна.

24.4. Расположение экранных форм

При работе с окнами экранных форм можно гибко задавать координаты их открытия. Координаты могут высчитываться относительно верхнего-левого угла единственного монитора или по абсолютным координатам многомониторной системы отображения.

Координаты открытия экранной формы

Чтобы указать точное место открытия окна в абсолютных координатах многомониторной системы:

› выставите свойство Положение окна в состояние **«Вручную»**;

› укажите свойства X окна и Y окна в абсолютных координатах многомониторной системы.

Монитор открытия экранной формы

Чтобы указать точное место открытия окна экранной формы на конкретном мониторе в многомониторной системе:

› выставите свойство Положение окна в состояние **«Вручную относительно монитора»**;

› укажите номер монитора в свойстве Монитор;

› укажите свойства X окна и Y окна относительно координат указанного монитора.

Позиция открытия экранной формы

Чтобы окно экранной формы открывалось по центру указанного монитора, родительского окна или родительской формы:

› выставите свойство экранной формы Положение окна в состояние **«По центру монитора»** / **«По центру родительского окна»** / **«По центру родительской формы»**;

› укажите номер монитора в свойстве Монитор, если был выбран режим **«По центру монитора»**.

ОБРАТИТЕ ВНИМАНИЕ

Если вместо многомониторной системы отображения используется единственный монитор, то во всех случаях значение свойства Монитор нужно оставлять равным **«0»**.

ПРИМЕЧАНИЕ

Чтобы окно экранной формы открывалось поверх остальных окон мнемосхемы, активируйте свойство Поверх всех окон.

Ограничение области перемещения экранной формы

Чтобы ограничить область перемещения экранной формы, воспользуйтесь соответствующей функцией.

SetBoundingRegion(int x, int y, uint width, uint height)

Определяет ограничивающую область для перемещения экранной формы. Экранную форму невозможно переместить за пределы ограничивающей области.

Входные параметры:

- x и y - координаты начала области ограничения;
- width и height - ширина и высота области ограничения, отсчитываются от точки начала области.

ПРИМЕР

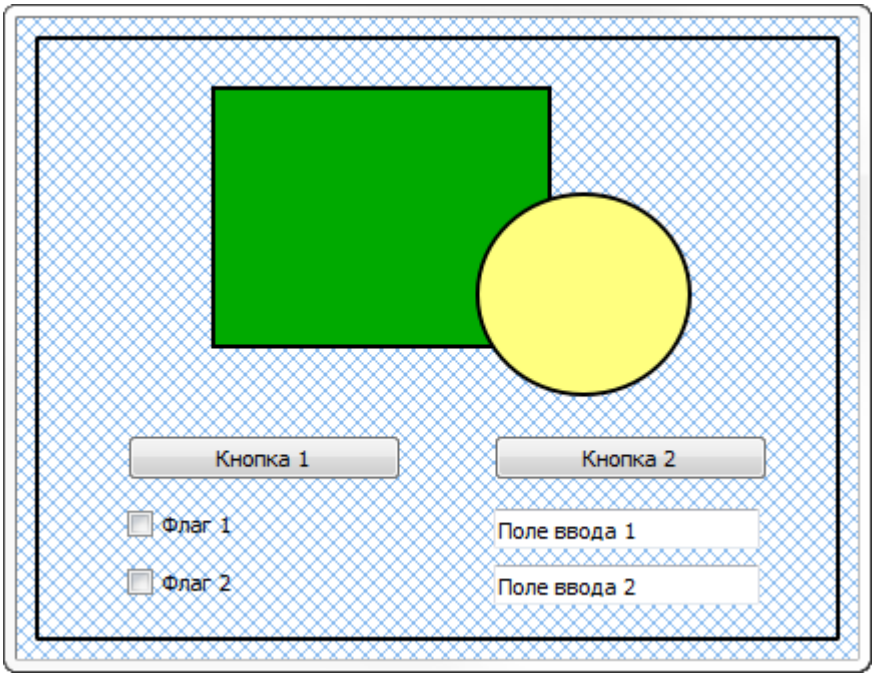
Установить ограничения перемещения окна

```
here.SetBoundingRegion(500, 200, 400, 400);
```

24.5. Скрытие заголовка окна и кнопок управления

Чтобы скрыть/отобразить кнопки управления окном (Свернуть, Развернуть, Закрыть), укажите значения соответствующим свойствам **Показать кнопку "свернуть"**, **Показать кнопку "развернуть"**, **Показать кнопку "закрыть"**.

Чтобы скрыть заголовок окна, и тем самым скрыть все кнопки управления окном и запретить пользователю захват окна для перемещения, укажите для свойства **Показывать заголовок** значение **«false»**.



24.6. Скрытие и деактивация объектов

Чтобы скрыть все визуальные объекты экранной формы в режиме исполнения, установите свойство экранной формы **Видимость** в значение **«false»**.

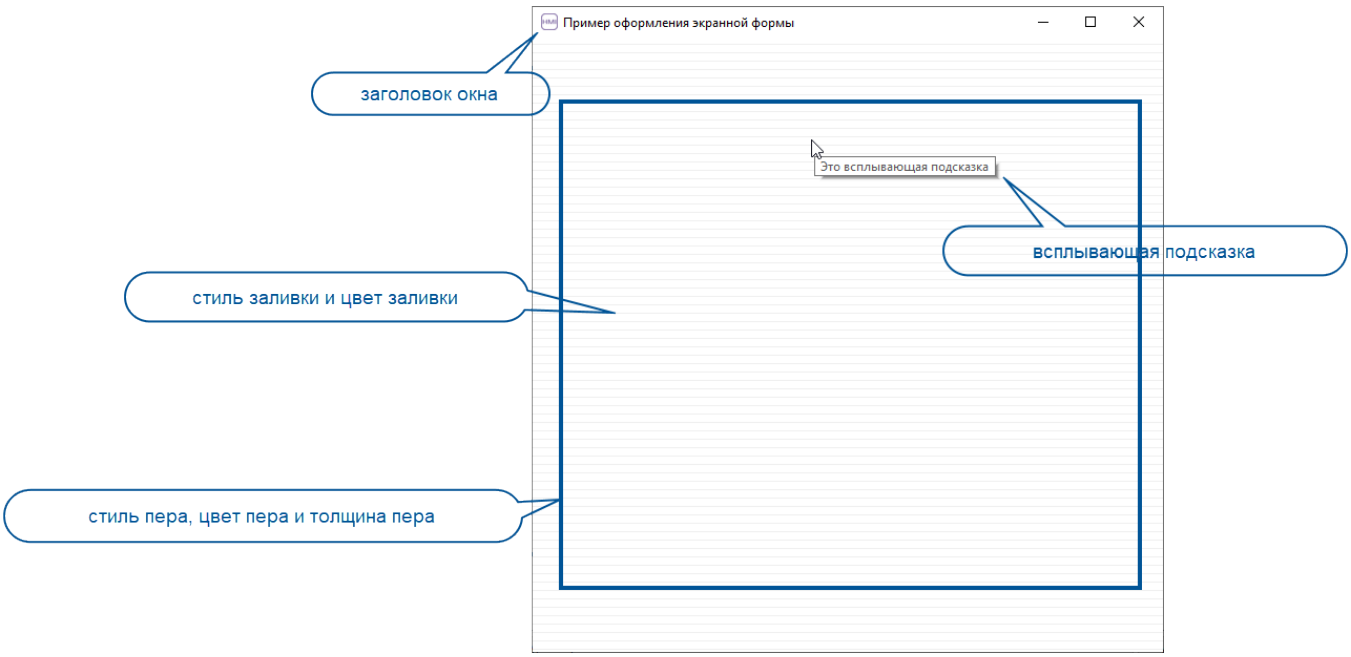
Чтобы деактивировать все объекты экранной формы (в режиме исполнения элементы будут видны, но не будут реагировать на действия пользователя), установите свойство **Включено** в значение **«false»**.

24.7. Скриншот формы

Чтобы сделать скриншот формы со всем её видимым содержимым, используйте функцию SaveScreenShot формы. Для просмотра подробного описания функции см. документ `SePlatform.HMI.Справочное руководство`.

24.8. Стиль и оформление экранных форм

Чтобы настроить оформление воспользуйтесь свойствами экранной формы в соответствии с поясняющим рисунком ниже.



Чтобы подробно ознакомиться со всеми свойствами, методами и событиями экранной формы см. документ SePlatform.HMI. Справочное руководство.

24.9. Открытие экранной формы во фрейме

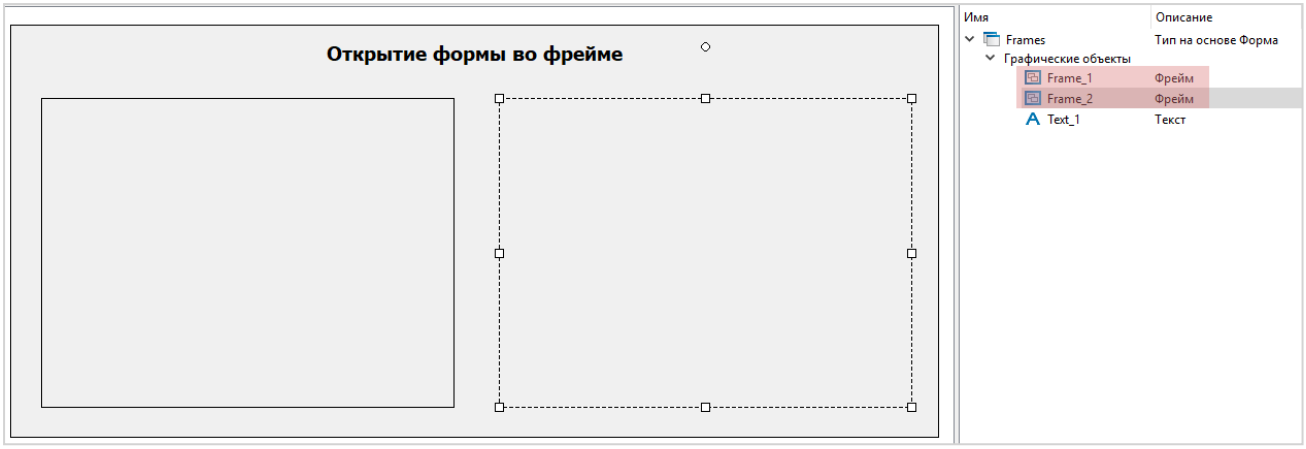
Чтобы содержимое экранной формы динамически открылось внутри другой формы, добавьте на экранную форму фреймовую область с помощью компонента **Фрейм**.

ПРИМЕР

Проект состоит из главной формы и нескольких зависимых форм, которые открываются в одной фреймовой области. Открытие экранных областей происходит по нажатию на специальные кнопки. Работа со всеми формами происходит только внутри главной формы.

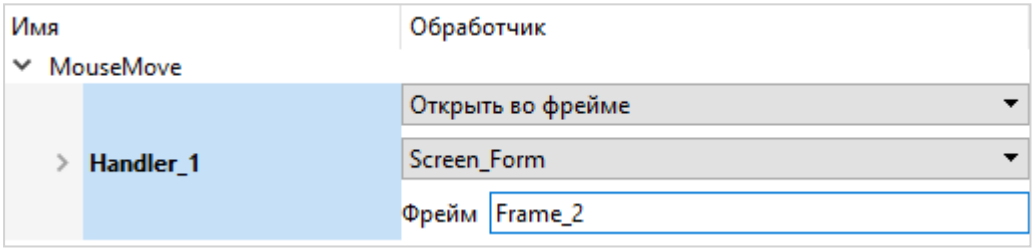
Сценарий внедрения:

1. Добавьте фреймовые области с помощью компонента **Фрейм**.

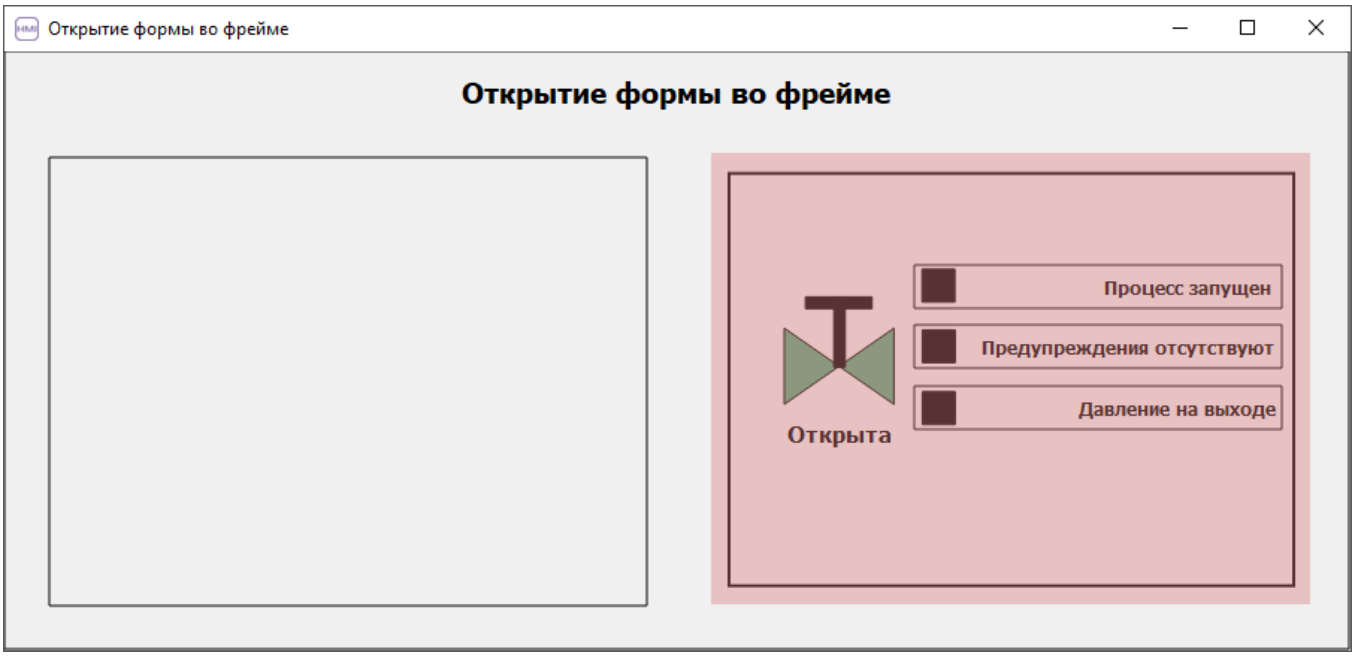


2. Создайте обработчик события, открывающий экранную форму во фреймовой области:

   - **Тип обработчика** - **«Открыть во фрейме»** (стр. 87);
   - **Тип формы** - из выпадающего списка выберите имя экранной формы, которая должна открыть во фрейме;
   - В текстовом поле введите имя фреймовой области, в котором будет открыта экранная форма.



При срабатывании события в указанной фреймовой области **«Frame_2»** откроется экранная форма **«Screen_Form»**.



24.10. Масштабирование и перемещение экранной формы во фрейме

Чтобы в режиме рантайма менять размер и двигать экранную форму во фрейме, используйте функции SePlatform.Om: ScaleContentTo, SetContentScale и MoveViewport. Функции работают при включенном свойстве фрейма Ручное управление масштабом.

Масштабирование формы во фрейме

Укажите размеры формы во входных аргументах функции ScaleContentTo или масштаб в аргументе функции SetContentScale. Если размер формы превысил размер фрейма, появляются полосы прокрутки. Убрать полосы прокрутки можно с помощью свойства фрейма Отображать полосы прокрутки.

Перемещение формы во фрейме

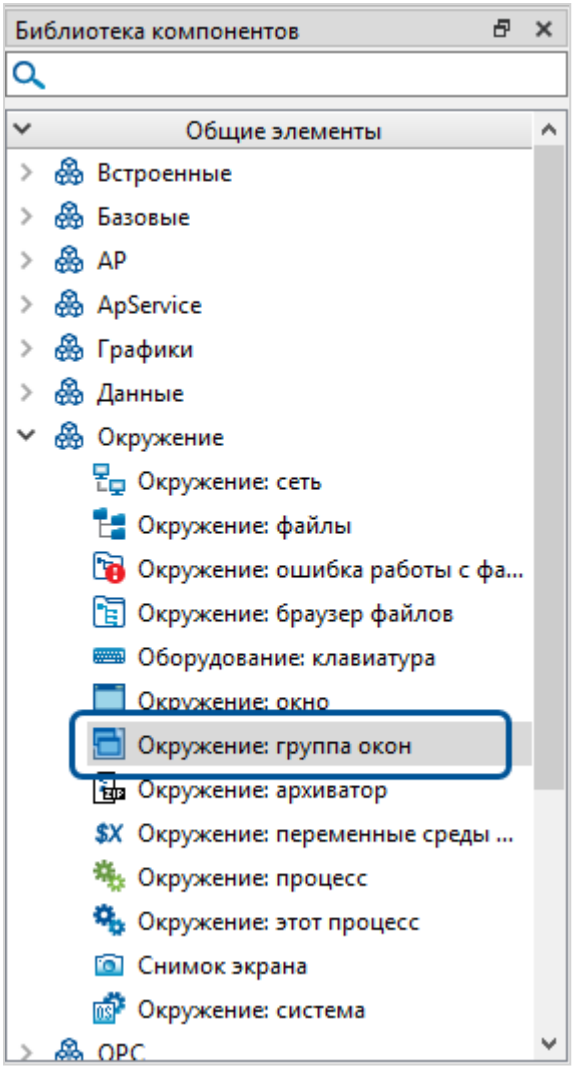
Перетаскивайте форму с помощью мыши или задайте координаты формы во входных аргументах функции MoveViewport. Координаты высчитываются относительно верхнего левого угла фрейма. Перетаскивать форму мышью можно при включенном свойстве фрейма Перемещение мышью.

24.11. Межоконное взаимодействие

Взаимодействие между основным и вспомогательными окнами осуществляется через компоненты окружения: **группа окон** и **окно**.

24.11.1. Группа окон

Чтобы взаимодействовать с окнами в группе, добавьте на экранную форму компонент **Окружение: группа окон**. Компонент невизуальный и виден только в области **Структура объекта**.



Компонент позволяет взаимодействовать с окнами внутри группы через свойства, функции и события (полный список содержится в справочнике).

События

Событие	Описание	Параметры
WindowOpened	Событие открытия нового окна в группе.	› windowID (тип string) - идентификатор окна в группе
WindowClosed	Событие закрытия окна в группе.	› windowID (тип string) - идентификатор окна в группе

Функции

Функция	Описание	Входные параметры
SendMessageByID	Отправляет сообщение конкретному окну, окно определяется по идентификатору.	› WindowID (тип int) - идентификатор окна, которому будет отправлено сообщение › MessageType (тип string) - тип сообщения, должен совпадать со значением типа в событии MessageReceived (компонент **Окружение: окно**) › MessageData (тип string) - тело сообщения, должно совпадать со значением тела в событии MessageReceived (компонент **Окружение: окно**)
SendToAll	Отправляет сообщение всем окнам, находящимся в группе.	› MessageType (тип string) - тип сообщения, должен совпадать со значением типа в событии MessageReceived (компонент **Окружение: окно**) › MessageData (тип string) - тело сообщения, должно совпадать со значением тела в событии MessageReceived (компонент **Окружение: окно**)

ПРИМЕР

Отправить сообщение на закрытие окна первому открытому окну

```
SendMessageByID(GetWindowIDbyIndex(0), "close", "Закрыть окно");
```

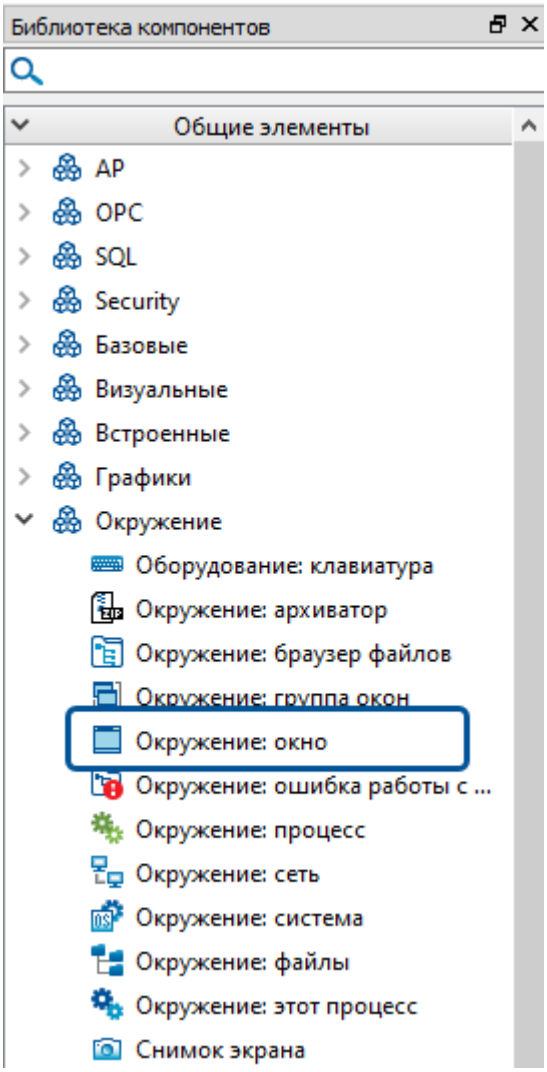
ПРИМЕР

Отправить сообщение на закрытие окна всем окнам

```
SendToAll("close", "Закрыть окно");
```

24.11.2. Окно

Чтобы настроить работу с оконным окружением, добавьте на экранную форму компонент **Окружение: окно**. Компонент невизуальный и виден только в области **Структура объекта**.



Компонент отвечает за все открытые экземпляры окон, входящих в одну группу окон. Один компонент **Окружение: окно** может содержать в себе несколько экземпляров окон. Количество экземпляров указывается в свойстве WindowCount компонента **Окружение: группа окон**.
Связь компонента с группой окон осуществляется с помощью свойства Группа окон. Действия, которые должны выполняться для окон в группе, прописываются в событии MessageReceived. Полный список свойств компонента см. в Справочнике.

Свойства

Свойство	Описание
Группа окон	Ссылка на компонент **Группа окон**, относительно которого будет происходить дальнейшая работа с компонентом. Указывается на вкладке **Редактор свойств**.

События

Событие	Описание	Параметры
MessageReceived	Событие получения сообщения.	› messageType (тип string) - тип сообщения. Значение параметра должно совпадать с типом сообщения в функции, отправляющей сообщение окну. › messageData (тип string) - тело сообщения. Значение параметра может использоваться для коммуникаций между окнами.

24.12. Навигация по иерархии экранных форм

Чтобы в режиме рантайма вывести имена всех или части экранных форм проекта в виде списка и оперативно переключаться между формами, предварительно сформируйте список с помощью элемента **Дерево**. Список может включать формы, объединенные одним признаком, а также повторять иерархию форм проекта. Для формирования списка используются функции SePlatform.Om. Для просмотра подробного описания функций см. документ `SePlatform.HMI.Справочное руководство`. Возможно ручное и автоматическое создание скрипта наполнения списка.

ОБРАТИТЕ ВНИМАНИЕ

Автоматическое создание скрипта наполнения дерева возможно только в проектах, интегрированных с SePlatform.Development Studio.

Для автоматического способа перейдите в контекстное меню элемента **Дерево** и выберите команду **Создать скрипт наполнения дерева**. Открывшееся окно позволяет выбрать формы, которые будут включены в список и указать данные, используемые при автоматическом создании скрипта.

Этапы создания скрипта

Подготовка столбцов

1. Укажите количество столбцов в свойстве **Число столбцов** для элемента **Дерево**.
2. Задайте имена столбцов с помощью функции **SetColumnName**. Для автоматического создания скрипта пропишите значения в строке **Имя столбца** и нажмите **Создать скрипт**.
3. Настройте видимость столбцов с помощью функции **SetColumnHidden**. Для автоматического создания скрипта установите флаги в строке **Скрыт** и нажмите **Создать скрипт**.
4. Укажите ширину столбцов с помощью функции **SetColumnWidth**.

Вставка нового элемента

1. Определите переменную, в которой будет id элемента - результат выполнения функции **AddItem**. Для вставки корневого элемента списка оставьте первый аргумент функции **AddItem** пустым. Для автоматического создания скрипта выберите формы, которые нужно добавить в список и нажмите **Создать скрипт**. В скрипте, созданном автоматически, учитывается иерархия форм в проекте.
2. Укажите имя элемента во втором аргументе функции **AddItem** либо с помощью функции **SetItemText**. Для автоматического создания скрипта выберите вид имени (**Name** - используется имя формы, **DisplayName** - используется значение свойства формы **Отображаемое имя**) в строке **Описание** и нажмите **Создать скрипт**.

В дополнение: задайте фон для элемента с помощью функции **SetItemBackgroundColor**, цвет текста элемента с помощью функции **SetItemTextColor** и установите иконку для элемента с помощью функции **SetItemIconPath**.

Вставка дочернего элемента

Для вставки элемента в конце списка дочерних элементов укажите id родительского элемента в первом аргументе функции **AddItem**. Для вставки элемента в точное место списка укажите id родительского элемента в первом аргументе функции **InsertItem** и порядковый номер дочернего элемента, после которого будет вставлен новый элемент, в третьем аргументе.

Привязка экранной формы к элементу

Для привязки формы к элементу списка укажите id элемента в первом аргументе функции SetItemData и id формы во втором аргументе. Для просмотра id формы перейдите в папку проекта и откройте файл нужной формы. В параметре **uuid** задан id формы.

Для автоматического создания скрипта выберите значение в строке **Данные** и нажмите **Создать скрипт**. Id форм будут подставлены автоматически.

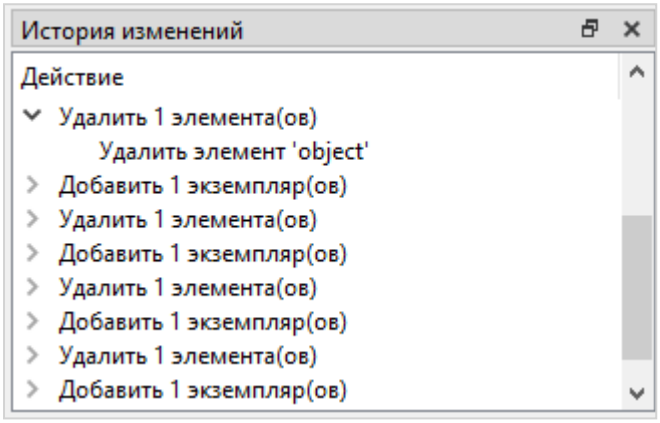
Очистка списка

Для удаления родительского элемента вместе с дочерними используйте функцию RemoveItem, для удаления всех дочерних элементов - функцию RemoveChildren.

25. Работа с историей изменений

История изменений - это область, отражающая в хронологическом порядке все изменения, произведенные пользователем с объектами на рабочей области. В области **История изменений** представлена информация о добавлении, удалении, переименовании, перемещении, изменении размера элемента и т.д. Каждый пункт истории можно раскрыть и детально просмотреть иерархию изменений (вплоть до информации по изменению конкретных атрибутов).

Жирным шрифтом выделяется последнее или активное изменение в истории.



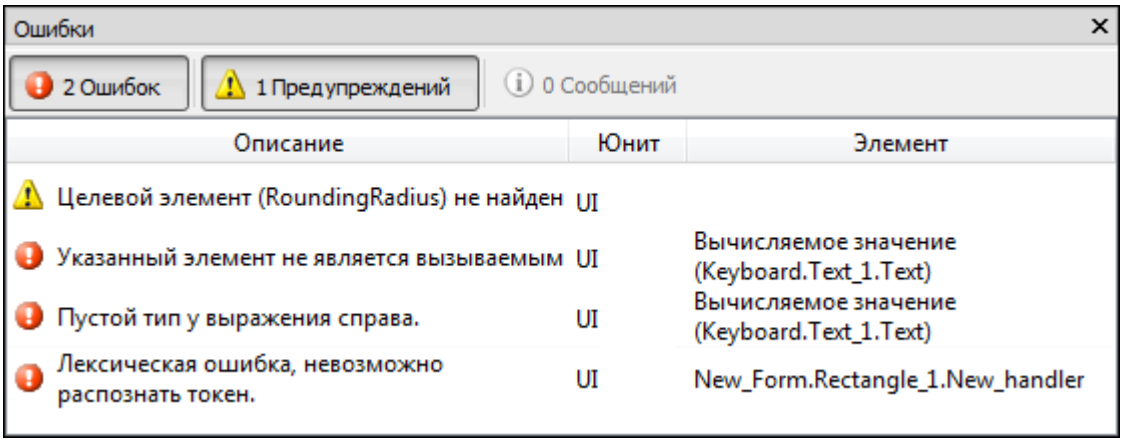
Чтобы отменить последнее действие, на панели инструментов нажмите кнопку **Отменить действие**  , выберите аналогичную команду в меню **Правка** или нажмите сочетание клавиш **«Ctrl»+«Z»**.

Чтобы вернуться к последнему отмененному действию, на панели инструментов нажмите кнопку **Повторить действие**  , выберите аналогичную команду в меню **Правка** или нажмите сочетание клавиш **«Ctrl»+«Y»**.

26. Просмотр ошибок

26.1. Просмотр ошибок в режиме разработки

Просмотреть ошибки, обнаруженные в ходе компиляции, а также предупреждения или сообщения можно в соответствующих вкладках области **Ошибки**.



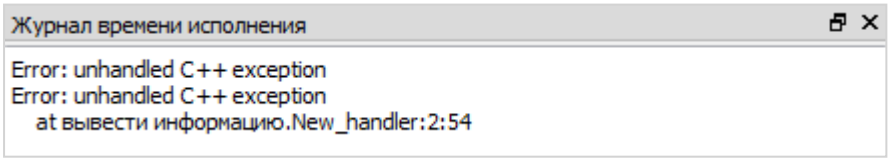
Чтобы перейти к месту ошибки в проекте, дважды щелкните по ошибке.

ОБРАТИТЕ ВНИМАНИЕ

Чтобы исправить ошибки некорректных ссылок или пустых присвоений (когда объект или свойство были удалены, а присвоения остались), используйте функцию **Проверка целостности**, которая находится в меню **Проект**.

26.2. Просмотр ошибок в режиме исполнения

Просмотреть ошибки, обнаруженные во время исполнения проекта, можно в области **Журнал времени исполнения**.



Чтобы в области **Журнал времени исполнения** выводились отладочные замечания во время исполнения кода, используйте элемент **Средство отладки**. Добавьте функцию Log элемента **Средство отладки** в нужных местах скрипта.

ПРИМЕР

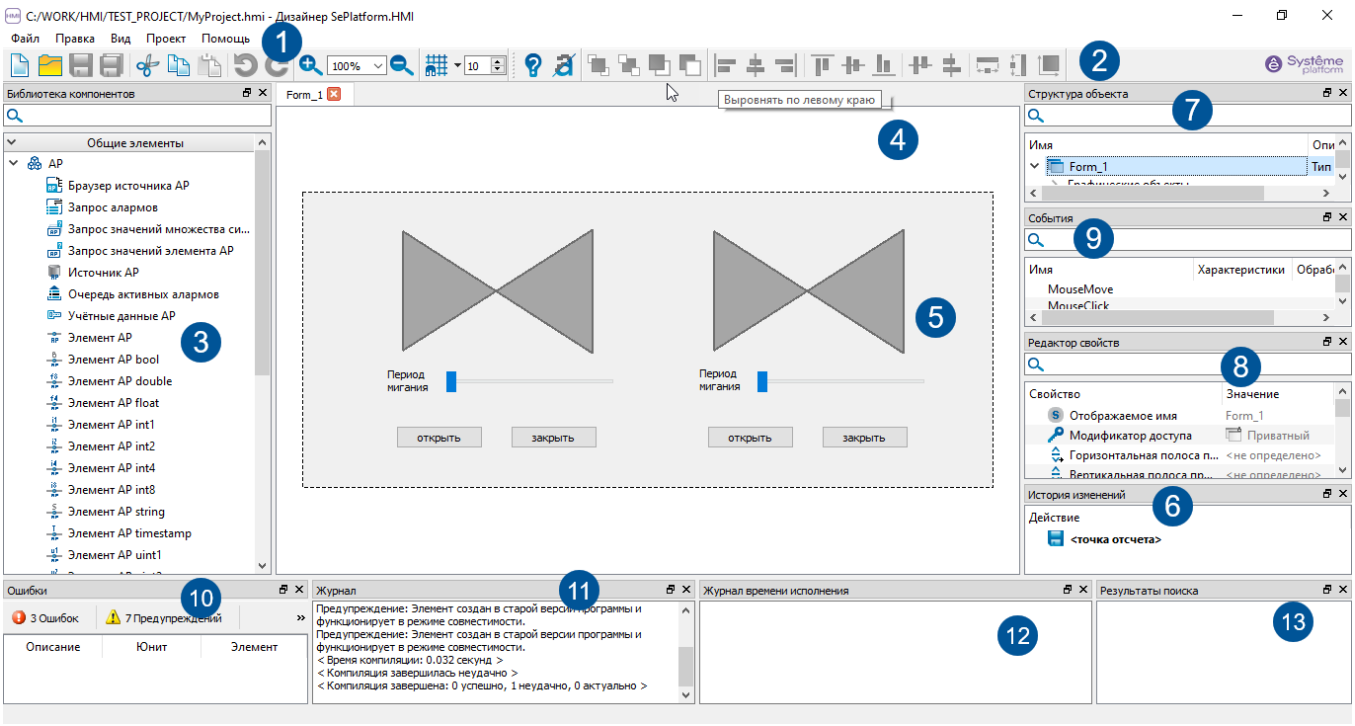
Отобразить сообщение в **Журнале времени исполнения** после окончания отладки

```
DebugTool.Log("Отладка завершена");
```

27. Приложения

Приложение A: Обзор среды разработки

Дизайнер SePlatform.HMI представляет собой среду разработки мнемосхем для проектов автоматизации технологических процессов. На рисунке ниже указаны основные области пользовательского интерфейса дизайнера SePlatform.HMI.



Описания областей:

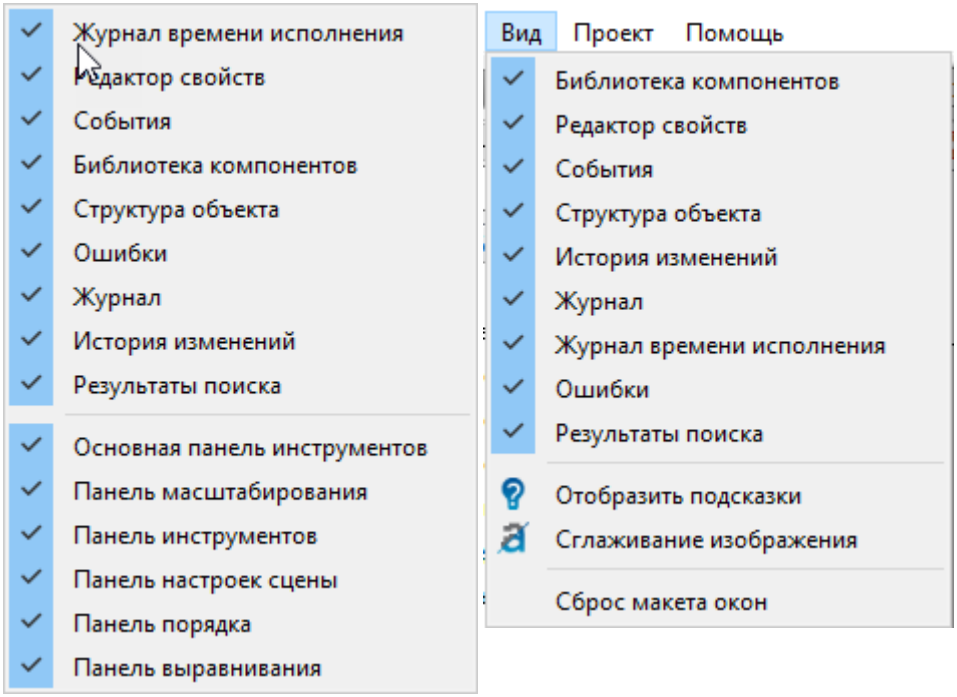
№	Описание
1	Панель меню
2	Панель инструментов содержит: › инструменты работы с проектом (стр. 14); › инструменты масштабирования (стр. 1) рабочей области; › инструменты форматирования (стр. 1) объектов; › инструменты выравнивания и распределения (стр. 23); › инструменты отмены/повтора действий (стр. 276).
3	Библиотека компонентов содержит набор стандартных компонентов для разработки проекта, а также экранные формы, глобальные объекты и типы (стр. 1), созданные пользователем.
4	Рабочая область - холст для размещения экранных форм и объектов.
5	Экранная форма, содержащая несколько графических объектов.
6	Область **История изменений** содержит лог изменений, проводимых с объектами рабочей области.

№	Описание
7	Область **Структура объекта** содержит иерархию объектов рабочей области.
8	Область **Свойства** содержит перечень свойств (стр. 66) выделенного объекта.
9	Область **События** содержит перечень событий (стр. 85) выделенного объекта.
10	Область **Ошибки** содержит перечень критических ошибок, предупреждений и сообщений, которые возникли в ходе компиляции проекта (стр. 15).
11	Область **Журнал** содержит лог событий, возникших в процессе компиляции проекта.
12	Область **Журнал времени исполнения** содержит события происходящие в процессе исполнения скомпилированного проекта (стр. 16).
13	Область **Результаты поиска** содержит перечень найденных объектов (стр. 29) по последнему запросу.
14	Область **Открытые вкладки** содержит список открытых в проекте вкладок

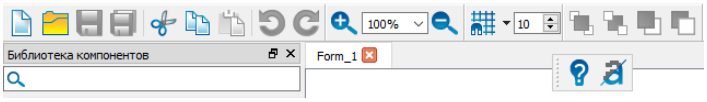
Приложение B: Настройка среды разработки

Дизайнер SePlatform.HMI позволяет пользователю гибко настраивать области пользовательского интерфейса. Области можно скрывать, отображать, перемещать внутри области Дизайнера SePlatform.HMI, выносить за пределы Дизайнера SePlatform.HMI, а также изменять размеры отображаемых областей. Настройки среды разработки сохраняются после перезапуска программы.

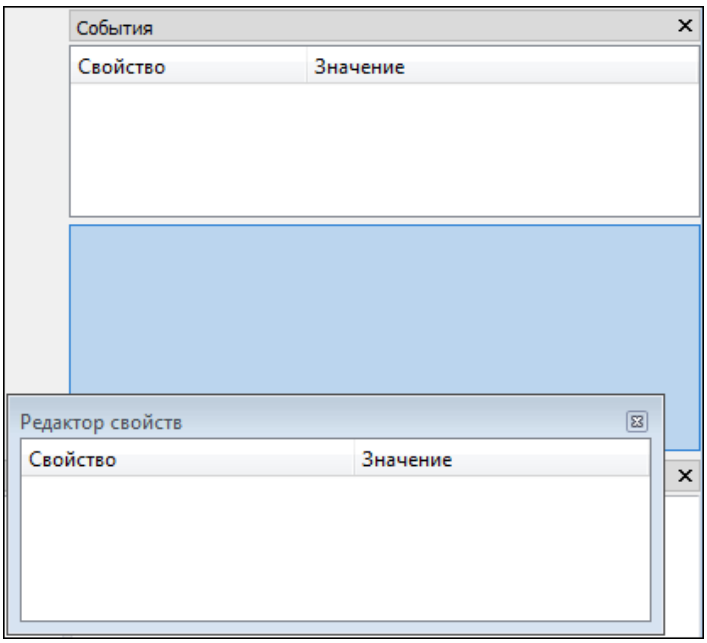
Чтобы скрыть/отобразить отдельные области пользовательского интерфейса, используйте флаги контекстного меню панели инструментов или опции меню **Вид**.



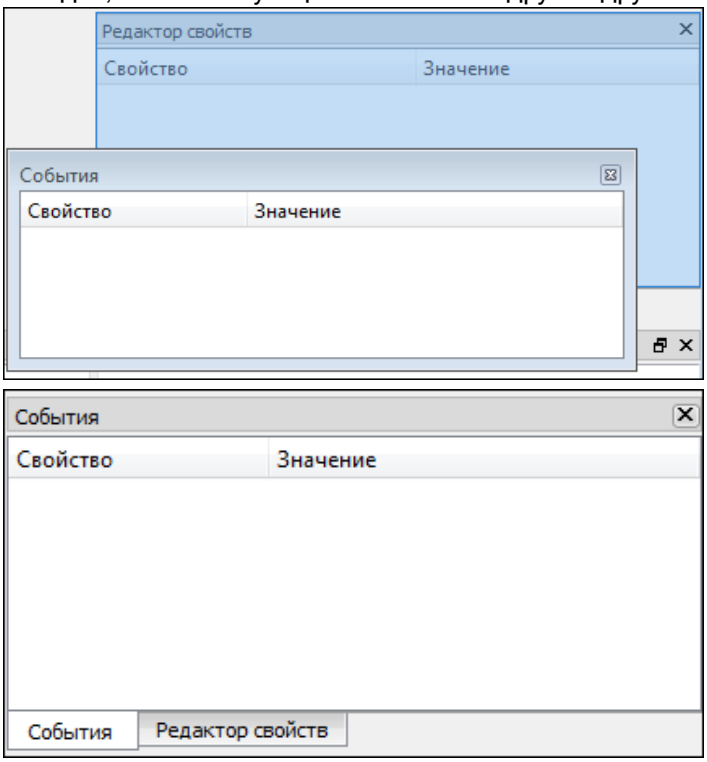
Отдельные составляющие панели инструментов можно перемещать в пределах/за пределами Дизайнера SePlatform.HMI и менять их размер.



Отдельные области пользовательского интерфейса (кроме панели меню) можно перемещать в пределах/за пределами Дизайнера SePlatform.HMI и менять их размер.



Чтобы создать области с вкладками, перетащите одну область на другую. Автоматически создадутся вкладки, соответствующие наложенным друг на друга областям интерфейса.



Приложение C: Лицензирование

Лицензионное использование программного продукта предполагает наличие электронного USB-ключа системы HASP. Для установки драйвера HASP-ключа запустите файл `HASPUserSetup.exe`, который можно загрузить по адресу: https://safenet-sentinel.ru/helpdesk/download-space/. В результате откроется окно **`Sentinel Runtime Setup`**. Для дальнейшей установки драйвера следуйте инструкциям мастера.

В случае отсутствия ключа или его изъятия из USB-порта во время работы, приложение будет работать в демонстрационном режиме.

Если лицензия отсутствует, то через 30 секунд после запуска проекта в рантайме, поверх экранной формы отобразится предупреждающее сообщение.

Приложение D: Горячие клавиши

Команда	Сочетание клавиш
Управление проектом	
Создать проект	«Ctrl»+«N»
Открыть проект	«Ctrl»+«O»
Сохранить изменения проекта на активной вкладке	«Ctrl»+«S»
Сохранить все изменения проекта	«Ctrl»+«Shift»+«S»
Общие команды	
Отменить последнее действие	«Ctrl»+«Z»
Повторить последнее действие	«Ctrl»+«Y»
Вырезать	«Ctrl»+«X»
Копировать	«Ctrl»+«C»
Вставить	«Ctrl»+«V»
Вставить строку со сдвигом в редакторе формул по условию	«Ctrl»+«Shift»+«+»
Вставка символа табуляции (сдвиг выделенного текста вправо)	«Tab»
Удаление символа табуляции (сдвиг выделенного текста влево)	«Shift»+«Tab»
Найти	«Ctrl»+«F»
Выделить всё	«Ctrl»+«A»
Закрыть текущую вкладку	«Ctrl»+«W»
Компиляция проекта и запуск в рантайме	
Запустить компиляцию проекта	«F5»
Запустить компиляцию активной вкладки	«Ctrl»+«F5»
Отменить компиляцию	«Ctrl»+«Break»
Запустить активную вкладку проекта в рантайме	«Ctrl»+«F9»
Запустить главную форму проекта в рантайме	«F9»
Порядок объектов на рабочей области	
Переместить объект на передний план	«Ctrl»+«L»
Переместить объект на задний план	«Ctrl»+«K»

Команда	**Сочетание клавиш**
Типы	
Перейти к типу	**«F12»**
Переименовать объект или тип	**«F2»**
Прочие	
Вызвать справку	**«F1»**

Список терминов и сокращений

HMI (Human-Machine Interface)	Графический интерфейс оператора.
IDE (Integrated Development Environment)	Интегрированная среда разработки.
OPC (OLE for Process Control)	Коммуникационный стандарт для компонентов систем автоматизации.